Hogyan szervezzünk programozási versenyt

Golda Bence

gbence@algernon.hu
ELTE IK

Absztrakt. Írásom a programozási versenyek szervezése során összegyűjtött tapasztalataimat tartalmazza, amelyek elsősorban a rám bízott informatikai infrastruktúrával kapcsolatos döntésekből és a végrehajtás lépéseiből állnak. Ez a gyűjtemény elsősorban azoknak a tanároknak és rendszergazdáknak szól, akik hozzám hasonlóan „a másik oldalon" vesznek részt a versenyeken.

1. Bevezetés

Több éves versenyzést követően, 2003 őszétől gyakorlatilag minden évben legalább egy országos vagy nemzetközi programozási verseny szervezésében vettem szervesen részt. Ezek között hagyományokkal rendelkező és presztízst jelentő versenyek is vannak, mint például a középiskolások körében lebonyolított Közép-Európai Informatikai Diákolimpia (CEOI-2005, Sárospatak[1]), illetve az ACM nemzetközi programozási versenye (Magyarországi ACM selejtező 2003–2007, ACM-CERC 2004–2006[2]), amely az egyik legöregebb és legnagyobb tömeget megmozgató verseny, a kategóriájában.

Ezekben a versenyekben közös, hogy közel azonos feltételekkel rendezik meg őket; viszonylag rövid idő alatt, sok, főként algoritmikusan nehéz, programozási feladatot kell megoldani, külső segítség nélkül, számítógépen, előre meghatározott programozási nyelvek segítségével. Ez versenyzői oldalról nagy előny, hiszen a hasonló versenyek hasonló kompetenciákat mérnek, ezért az egyikre való felkészülés, egyben a másikra való felkészülést is jelenti. Ugyanezen okok miatt a versenyek lebonyolítását is könnyebb, hiszen ha a szervezők megosztják egymással tapasztalataikat, a feltételek hasonlósága miatt, a felmerülő problémák és az azokra adható, minden igényt kielégítő válaszok és reakciók is rendkívül hasonlók.

Írásom célja a programozási versenyek rendezésével kapcsolatos tapasztalataim összegyűjtése, illetve az, hogy végre ne csak öröklés és szájhagyomány útján szálljon a tudás szervezőről szervezőre, hanem a tudomány is segítse munkánkat.

2. Programozási versenyek és jellemzőik

A nagy és ismert programozási versenyek az elmúlt 20 évben gyakorlatilag nem változtak, legfeljebb csak annyiban, hogy a hardveres és szoftveres fejlődést követve új programozási nyelvek ismeretével is lehet indulni az egyre bonyolultabb feladatokat és problémákat felsorakoztató megmérettetéseken. Ide tartoznak a országos és nemzetközi programozási tanulmányi versenyek (*Logo OSzTV*[3], *Nemes Tihamér*[4], *OKTV*, *CEOI*[5] és *IOI*[6]). Továbbá azok, amelyeken elsősorban a versenyzők algoritmizáló és matematikai képességeit szeretnék mérni, a programozás eszközének segítségével (*ACM ICPC*[7], *ICFP*[8]).

Azonban nem szabad megfeledkeznünk a játékos küzdelemként induló, később komoly versenyzői bázist begyűjtő versenyekről sem, mint például az „Interaktív Imagine Mesekészítés" Verseny, illetve a külföldön is ismert *Hoshimi-project*[9] vagy a *Core Wars*[10]. A versenyek

sorát a technikai újításokat és új informatikai paradigmákat felsorakoztatók zárják, amelyek mögött legtöbbször támogató nagyvállalatok állnak, ilyen a *TopCoder*[11] és *ImagineCup*[12].

Ezeken kívül számos informatikához közel álló verseny van még mind itthon, mind külföldön, azonban ezek elsősorban nem a programozási nyelvek köré épülnek, így a C3 Alapítvány által szervezett „*<19 Szabadfogású Számítógép Verseny*"[13] és a hasonló megmérettetések nem képezik kutatásaim részét.

Mivel elsősorban egy konkrét versenytípus szervezése során gyűjtöttem össze tapasztalataimat, ismertetem azokat a jellemzőket és tulajdonságokat, amelyek esetén a leírtak segíthetik a szervezők munkáját:

- Formátum: *papíros* vagy *számítógépes*. A versenyek azon szakaszaiban, amelyek nehezen automatizálhatók – például a programkód „szépségének" mérésekor, amelyre később még visszatérek – szokás papíros formában összegyűjteni a megoldásokat. A „kézzel és szemmel" történő kiértékelés miatt ez rendkívül erőforrásigényes módja a javításnak, ezért – egy átlagos egyetemi versenyen – kevés szervező és sok versenyző esetén kivitelezhetetlen, sőt objektivitása is vitatható. A *számítógépes* versenyek előnye ezzel szemben az, hogy minden automatizálható feladatot – programok ellenőrzése, feladatok kiosztása, stb. – a számítógépek felügyelet nélkül képesek elvégezni, így gyakorlatilag konstans számú szervező elég egy nagyméretű programozási verseny technikai lebonyolítására.

  Mivel a programozás alapvetően számítógépet igénylő feladat, gép nélkül a megoldás és az ellenőrzés is erőltetett és nehézkes, napjainkban kevéssé indokolt, ezt követően csak a *számítógépes* versenytípusokkal fogok foglalkozni.

- Segítség: *zárt*, *open-book*, *open-net*, vagy *teljesen szabad*. A számítógépes feladatmegoldásnak van egy kellemetlen mellékhatása is, a versenyzők a gép – gyakorlatilag – összes erőforrásához hozzáférhetnek, beleértve a számítógépen található dokumentációt is.[14] Emiatt a számítógépes programozási versenyeken a lexikális tudás mérése értelmetlen. A teljesen *zárt* versenyek ezért az utóbbi időben a számítógépen található dokumentáció használatát engedélyezik.

  A felhasználható segítség szempontjából a másik véglet a *teljesen szabad* verseny, ahol bármilyen külső segédeszköz felhasználható – a legtöbb esetben külső emberi erőforrás azonban nem. A két véglet között helyezkednek el az *open-book* és *open-net* versenyek, amelyek a felhasználható tudás minőségét nem, csak a mennyiségét korlátozzák – egy jó könyv vagy tartalmas honlap sokkal nagyobb segítség lehet, mint 100 silány.[1]

- Kommunikáció: *tiltott*, *limitált* és *engedélyezett*. A tiltott és engedélyezett kommunikáció között fél úton található *limitált* alatt azt értjük, amikor a versenyzők a szervezőknek (feladatok kitalálói, bírók, stb.) kérdéseket tehetnek fel és ezekre egy szavas vagy nagyon rövid válaszokat kapnak – ebbe beletartozik a verseny során kiértékelt programokra adott válasz is, például „fordítási hiba" vagy „helytelen kimeneti formátum".[15]

[1] Szerencsére 10-nél több könyvet nehéz elcipelni a versenyre, ezért a résztvevők rá vannak kényszerítve a hatékony válogatásra.

- *Önálló* vagy *csoportos* versenyek. Szűkebb értelemben véve ez a csoportmunka versenyen belüli megjelenésének legfontosabb meghatározója. Amennyiben önálló versenyzők vagy kis csoportok (2-3 versenyző) vesznek részt a versenyen, a megoldásokat leginkább individuumként állítják elő a versenyzők, így a feladatok összehangolására nincs szükség. Nagyobb csoportok és főleg hosszabb idejű, nagyobb léptékű feladatok esetén, azonban megjelenik a csoportot irányító „menedzser"[16], akinek munkája és eredményei a verseny mérőszámai között egy új dimenzióban jelenik meg.

- *Versenyzők száma*. A verseny olyan mérőszáma a *versenyzők száma*, amely a versenyzők felkészülési stratégiájára nincs befolyással, azonban a szervezők számára az egyik legfontosabb jellemző. *Kevés* versenyző esetén 1-2 tanár is képes lebonyolítani a versenyt, *nagyméretű verseny* megszervezéséhez azonban már komoly, összehangolt csapatmunkára van szükség.

- Időtartam: *kötött idejű* vagy *kötetlen idejű* (a hosszú időre adott feladatok is ilyenek). Míg az utóbbit leginkább a gyakorlások, kötetlen versenyek, felkészítő oktatások során szokás alkalmazni, a rangos megmérettetések mindegyike *kötött idejű*, azaz rövid idő alatt kell megoldani a feladatokat. A verseny időtartamának és a feladatok számának és nehézségi fokának helyes megválasztásával elérhető, hogy a mezőny – a verseny céljainak megfelelően – lehetőleg egyenletes eredményt produkáljon.

- *Nehézség*. Talán az egyik legfontosabb tényező a versenyzők és a verseny megítélése szempontjából, azonban a versenyek technikai lebonyolítása szempontjából gyakorlatilag meg sem jelenő jellemző,[2] ugyanis egy 1000 versenyzős „Hello World!" nehézségű verseny ugyanannyi szervezést igényel, mint egy 1000 versenyzős komoly algoritmikus és programozói tudást követelő megmérettetés.

  A leendő szervezők figyelmébe ajánlom, hogy tapasztalat szerint, a diákok akkor mennek haza boldogan egy versenyről, ha van olyan versenyző, aki (majdnem) minden példát megold, és nincs olyan, aki egy példát sem tudott megoldani.

- *Algoritmus-*, vagy *feladat-orientált* versenytípus. Míg az algoritmus orientált versenyek esetében konkrét algoritmusok köré épülnek fel a feladatok, és a megoldás során leginkább azok kitalálása a verseny legnehezebb része, *feladat orientált* versenyeken a legtöbbször a való életből kiragadott problémákra adandó minden igényt kielégítő megoldás leprogramozása jelenti a nehézséget. Más tudományterületről vett példával élve az előbbi inkább egy laboratóriumi környezetre, a második pedig egy betegekkel teli kórházra hasonlít.

- Ellenőrzés: *forráskód* vagy *kimenet*. A programozási versenyek egyik nagy előnye a többi tudományos versennyel szemben, hogy viszonylag egyszerű, automatizálható az eredmények értékelése. A versenyzők szemszögéből azonban nem mindegy, hogy megoldásukat „szépen" kell prezentálniuk – tehát mindenképp szükség van a forráskódra a kiértékeléshez, – vagy elég, ha csak „működik" a program.

[2] Volt olyan az évek során, hogy – az informatikai infrastruktúráért felelős szervezőként – a verseny vége után pár nappal láttam először a feladatokat.

- *Azonnal kiértékelt* vagy a *verseny befejezését követően egyszer ellenőrzött* versenytípus. A versenyzők szempontjából az azonnal kiértékelt megoldások nagy segítséget jelentenek a verseny során, hiszen így a résztvevők azonnal visszajelzést kapnak programjuk helyes vagy helytelen működéséről, amelyet javítani tudnak. Nem szabad megfeledkeznünk arról sem, hogy így a verseny közben felmerülő, beadott megoldásokkal kapcsolatos bizonytalanságuk is kevésbé hangsúlyos, nem növeli a versenydrukkot.

  Egyedül a szervezők szemszögéből előnytelen versenytípus az azonnal kiértékelt, ugyanis az egyik legkomolyabb megmérettetés megbízható kiértékelő rendszert, illetve a feladatokhoz illő ellenőrző programokat találni és készíteni.

Az ismertetett tulajdonságoknak megfelelően az ACM versenyek besorolása – amelyek szervezéséből a legtöbb tapasztalatot merítettem: *számítógépes, azonnal kiértékelt, open-book, limitált kommunikációt engedélyező, kis csoportos, kötött idejű, forráskódot igénylő, nehéz, algoritmus orientált* verseny, amelyen átlagosan 100-200 versenyző vesz részt egy regionális forduló alkalmával. Az itt leírt tapasztalatok egy ehhez hasonló verseny szervezése során nyújthatnak nagy segítséget.

3. Előkészületek

Egy programozási verseny szervezése három részből áll, az előkészületekből, a versenyből és az utómunkálatokból. A legtöbb időt és energiát felemésztő előkészületek akkor fejeződnek be, amikor az első versenyző belép a verseny helyszínére és leül a számítógépe elé – online esetben belép a verseny weboldalára.

A tennivalók azon halmazára fogok most koncentrálni, amelyek a verseny megrendezésének szükséges feltételei, és amelyeknek az eredménye már az előkészületek alatt is kézzel fogható.

3.1. Informatikai infrastruktúra összeállítása

Egy verseny alapvető jellemzője, hogy minden résztvevő egyenlő feltételek mellett vághat bele a versenybe. Informatikával kapcsolatos versenyeken, különösen ahol a számítógép szerves részét képezi a versenynek, az egyenlő feltételek biztosítása nem triviálisan megoldható feladat.

3.1.1. Hardver

Függetlenül attól, hogy a verseny eredményeinek kiértékelése hol történik – központi szerveren vagy már a versenyző gépén megszületik az eredmény, – ha a versenyzők gépei között (nagy) különbségek vannak, a versenyzők egy része mindenképp hátrányos helyzetbe kerül.

1. Sebesség és memória: programozási feladatok megoldása a gép szempontjából általában két-három nagyobb lépésből áll, a programkód beviteléből, a fordításból és opcionális esetben – ha a versenyző követi az általános fejlesztési lépéseket – a tesztelésből.

   Ezek közül a programszerkesztés kevésbé, a másik két lépés viszont igencsak erőforrásigényes feladat. Egy lassabb, kevesebb memóriával rendelkező gép a fordítások és tesztelések miatt többlet időt halmozhat fel a gyorsabbakhoz képest,

ami a versenyző számára értékes – gondolkodási időből elvesztegetett – perceket jelent.

Ha nincs mód azonos számítógépek beszerzésére, jó megoldást jelenthet a gyorsabb gép sebességének korlátozása, illetve a felhasználók által foglalható összes memória csökkentése. Linux operációs rendszer alatt ezt az *acpi* kernel modul és a felhasználói limitek (*ulimit*) megfelelő beállításával lehet elérni.

Amennyiben erre sincs lehetőség, vagy nagyon különbözőek a számítógépek, központi számítógép felállításával is biztosíthatók az egyenlő feltételek, azonban ügyelni kell arra, hogy a versenyzők ne foglalhassák le egymás elől a számítógép erőforrásait.

2. *Képernyő*: a TFT és LCD képernyők miatt manapság már nem, de 2-3 éve még gondot okozott a monitorok megfelelő frekvenciájának beállítása. Egy rosszul beállított, „villogó" monitor, egy 5 órás verseny alatt rendkívüli fejfájást tud okozni, ami a verseny alapvető szükséges feltételeinek mond ellent.

   Az előző ponthoz rendkívül hasonló feltételt jelent a képernyők felbontásának egységesítése, ugyanis egy nagyobb felbontású képernyőn négyzetes arányban több információt lehet megjeleníteni, így például egy csoportos verseny esetén (pl. ACM) egy nagy felbontású képernyőn megtehetik a versenyzők, hogy amíg az egyik gépel a másik kódot elemez. Kisebb felbontás esetén ez nehézkes, ami hátrányt jelenthet az adott csapatoknak.

   Amennyiben nem megoldható az azonos méretű monitorok biztosítása, az egyes gépeken elérhető lehetséges felbontások maximumainak minimumát kell beállítani az egyes gépeken – és nem szabad megengedni a képernyő felbontásának megváltoztatását.

3. *Billentyűzet (és egér):* mindenki szeret a lehető legkényelmesebb környezetben dolgozni, így nem meglepő, hogy nem volt még eddig olyan általam szervezett verseny, amelyre ne hozott volna valaki saját billentyűzetet.

   A saját billentyűzet engedélyezésének azonban van három következménye. Az egyik a számítógépes hardver egységének megbontása – amely ellen leginkább a számítógépek helyi rendszergazdái szoktak kardoskodni, érthető okokból, a második pedig a billentyűzet kompatibilitásának problémája, ugyanis elképzelhető, hogy a számítógép a verseny alapvető szoftvereivel nem támogatja a speciális billentyűzeteket (pl. USB csatlakozósokat).

   A harmadik következmény már nem ennyire szembetűnő, mégis potenciális problémát jelenthet. A *zárt* és *open-book* versenyeken a versenyző csak a tudására és a könyveire támaszkodhat, elektronikus forrásból származó segítség – a gépen találhatókon kívül – nem megengedett. Ezeken a versenyeken értékes másodperceket jelenthet, ha a kezdéskor már a versenyző rendelkezésére áll egy előre megírt saját programkód (vagy váz), amely gyakran használt kódrészleteket, előre begépelt, optimalizált, általános problémákat megoldó algoritmusokat tartalmaz.

   Az előnyszerzés lényege, hogy egy billentyűzetbe épített speciális célhardverrel, a versenyző, a számítógép bemenetére küldheti az előre beégetett programkódokat –

ami gyakorlatilag detektálhatatlan, hiszen pontosan ugyanúgy kerül a gépbe a programkód, mintha begépelték volna.

Ezen indokok mellett minden versenyen külön érdemes mérlegelni, hogy a versenyzők kényelmét vagy a veszélyeket tartjuk fontosabbnak, ennek megfelelően érdemes engedni vagy tiltani saját perifériák használatát.

4. *Nyomtató* – ideális esetben a számítógéphez tartozó perifériák közvetlenül a számítógép mellett helyezkednek el. A nyomtató esete azonban speciális, ugyanis lényegesen kisebb a szerepe a verseny során, a fent említettekhez képest.

   Hálózatba kötött számítógépek esetén, az erőforrások takarékos felhasználása miatt, egy, de mindenesetre kevés számú, nagy teljesítményű nyomtatót szokás a résztvevők rendelkezésére bocsátani a verseny ideje alatt. Amennyiben a nyomtató vagy nyomtatók nincsenek helyhez kötve, érdemes helyüket úgy megválasztani, hogy lehetőleg a nyomtatást követő 1 percen belül, a szervezők segítségével, a versenyzők kézhez kapják nyomtatott dokumentumaikat.

   Ezzel megelőzhetjük, hogy a versenyzők egymást kerülgessék a nyomtató és a nyomtatott anyagok körül.

A versennyel kapcsolatos hardveres igények és változtatások a legidőigényesebbek és legköltségesebbek. Érdemes ezért a szervezőknek előre eldönteniük, hogy hol húzzák meg a versenyzők igényeinek kielégítése és a ráfordított idő között húzódó határvonalat.

A teljesség kedvéért érdemes még egy lehetőséget is számba venni, a vékony klienseket alkalmazó architektúrát. Egy ilyen felépítésű rendszerben a központi számítógépre csatlakoztatott kis erőforrással rendelkező terminálok jelentik a versenyzők számára a feladatmegoldás eszközét – amely gyakorlatilag csak egy monitort és egy billentyűzetet jelent, hiszen az érdemi munkát a központi gép végzi.

Az egyenlő esélyeket biztosító architektúra előnye egyben a hátránya is, hiszen egymástól veszik el a versenyzők a szabad erőforrásokat. Ha valamilyen oknál fogva mégis ilyen rendszer segítségével bonyolítunk le egy programozási versenyt, a felhasználható erőforrások (memória, processzoridő, stb.) maximumát érdemes körültekintően beállítani.

A közös gép legnagyobb hátránya ellen azonban még így sem védekezhetünk; hiba esetén valószínűleg az összes versenyző versenyének egyszerre van vége. Érdemes ezért ezt a lehetőséget – ha tehetjük – már a verseny tervezésének első fázisaiban elvetni.

3.1.2. Szoftver

A versenyzők számítógépeinek összeállítása során a második lépés a gép szoftvereinek telepítése. Az egyenlő esélyek ebben az esetben is csak úgy biztosíthatók, ha a gépek hardverein túl a szoftverek is minden gépen azonosak.

Részletes listát a szoftverekről a folyamatosan változó verziók, és az éppen aktuálisan divatos alkalmazások miatt értelmetlen összeállítani, azonban a programozásra széles körben használt alkalmazások típusai egyszerűen meghatározhatók:

- *Fejlesztői környezet* : modern programozási nyelvekhez manapság kész keretrendszereket készítenek, amelyek a programszerkesztő funkciókon kívül automatikus fordítási és tesztelési feladatokat is képesek ellátni. Bár ezeket

leginkább ipari környezetben alkalmazzák, sok tanár – az ilyen irányú ismereteiket alapul véve – ezek segítségével ismerteti meg a diákokkal, versenyzőkkel a programozást, így számukra ezek az eszközök jelentik a kényelmes, megszokott környezetet.

- Fel kell készülnünk azonban arra is, hogy vannak (és lesznek) olyan versenyzők, akik nem szeretik a komplex fejlesztői környezeteket, hanem jobban érzik magukat a már megszokott – és minden más probléma esetén is jól működő és bevált – alkalmazásaikkal. Ilyen esetek miatt mindenképp fel kell készíteni a versenyzői számítógépeket a következő szoftver típusok, lehetőleg minél szélesebb tárházával:

    a. *programkódszerkesztő* – pl. VIM, Emacs, mcedit, stb.

    b. *fordítóprogramok* – a versenyen használt programozási nyelvek fordításához és teszteléséhez szükséges programok, pl. GCC/G++, FreePascal, stb.

    c. szkriptezési lehetőség – a fordítás és tesztelés fázisait a versenyzők általában kézzel végzik, azonban előfordulhat, hogy nagyobb vagy bonyolultabb eseteknél cél- és feladat orientált programok segítségével könnyebb a megoldások ellenőrzése. Ilyen programok írására a népszerű szkriptnyelvek a legalkalmasabbak – pl. Shell-nyelvek, Ruby, Python, Perl, PHP, stb.

    d. *kalkulátor* – a példákhoz gyártott tesztesetek gyakran nagy számokat tartalmaznak, amelyeket csak számológéppel lehet kiszámítani. Mivel elektronikus eszközök használata számos versenyen nem megengedett (pl. ACM ICPC, CEOI, IOI, stb.) egy kalkulátor program a számítógépen rendkívül nagy segítséget nyújthat, gyakorlatilag alapkövetelmény.

    e. *mérőeszközök és hibakereső programok*: amennyiben a versenyzők számára ismertek a feladatmegoldásaikkal szembeni elvárások (idő-, memóriafelhasználás, stb.), a limiteket mérni képes mérőeszközök nagy segítséget nyújthatnak a fejlesztés során. (pl. ddd, gdb, stb.)

- *Időszinkronizáló* és *-kijelző*: a verseny feltételeinek kihirdetésébe beletartozik a verseny során felhasználható idő is. Ez más esetekben egy időtartamot jelent, aminek az elejét például startpisztollyal jelzik, számítógépes versenyeken azonban lehetőségünk van a gépeket a saját céljainknak megfelelően alkalmazni az időintervallum kezdetének és végének pontos prezentálására.

  Érdemes tehát egy központi szerverrel szinkronban tartani a számítógépek óráit, így a versenyekre jellemző „utolsó percek hajrájában” is egyenlő esélyeket biztosíthatunk.

- *Böngésző* : mind a gépen található dokumentáció, mind a versenyt lebonyolító központi szoftver elérésének legegyszerűbb és legkézenfekvőbb módja – manapság – egy grafikus böngésző alkalmazása. Ezek a programok a dokumentáció böngészésén kívül képesek a versenyfeladatok prezentálására és megfelelő háttérrendszer segítségével a megoldások beszedésére, illetve az eredmények prezentálására is.

Előnyük bármilyen más megoldással szemben, tudásuk mellett, hogy gyakorlatilag mindenféle operációs rendszeren és környezetben hasonlóan működnek, ezért a versenyzők nagy része biztosan rutinosan tudja majd kezelni a programot – így nem kell a versenyen a feladatokon túl még azzal is megküzdeniük, hogy a versenyt lebonyolító szoftver pontosan hogyan működik.

Ez utóbbi szempont a legfontosabb a versenyen használt számítógépek összeállítása során, a versenyzőknek a verseny során a versenyzés a legfontosabb feladatuk. Amennyiben a környezet gördít eléjük akadályokat, a feladatukra kevésbé tudnak koncentrálni, ami a versenyzést hátráltatja.

Hasonlóan az eddigiekhez az egyenlő feltételek biztosítása miatt szükségesek a következő szoftveres biztonsági intézkedések a versenyzők által használt számítógépeken:

1. *Erőforrások korlátozása* : a lebonyolítás során csak és kizárólag azokat az erőforrásokat biztosítsuk a versenyzők számára, amelyekről tudjuk, hogy szükségesek a megoldás előállításához. A „tételezzük fel a legrosszabbat” elv alkalmazására azért van szükség, mert egy szervezők által nem ismert program vagy erőforrás alkalmazása egyes versenyzőket előre kiszámíthatatlan előnyhöz juttathat. Így ha például nincs szükség az Internetre a feladatok megoldása során, érdemes mindent megtennünk annak érdekében, hogy senki ne is használhassa azt a verseny időtartama alatt.

2. *Csak rendszergazda* felhasználó tudjon alapvető hardveres vagy szoftveres módosításokat végrehajtani a számítógépen. Ennek egyszerű oka abban keresendő, hogy a korábban említett korlátozó beállítások megkerülhetők vagy átugorhatók, ha a versenyzők korlátlan hozzáférést kapnak a számítógép erőforrásaihoz.

   Például semmit sem ér egy hálózatot letiltó tűzfalszabály vagy program, ha kiiktatható vagy megkerülhető, sőt, ebben az esetben – amennyiben olcsó eszközökkel vannak összekötve a számítógépek – az azonos alhálózatba kötött gépek és az előttük ülő versenyzők egyszerűen kommunikálhatnak egymással.

3. *Monitorozás* : a versenyben rögzített szabályok betartása a leghatékonyabban akkor biztosítható, ha széleskörű informatikai monitoring rendszerrel figyeljük és ellenőrizzük a teljes rendszert és a versenyzők rendszerben végrehajtott akcióit.

   Nem szabad azonban elfelejtenünk azt, hogy egészen addig, amíg egy versenyzőről be nem bizonyosodik, hogy csal, átlagos versenyzőként kell kezelnünk, így nem mehet a megfigyelés a verseny rovására – például a monitorozó program nem lassíthatja számottevően a megfigyelt gépet.

3.1.3. Telepítés

Technikai szervezőként, a versenyre való felkészülések során, minden esetben a legtöbb energiámat és időmet a versenyzők számítógépeinek felkészítése emésztette fel, amely jól jellemzi, a feladat – általam becsült – fontosságát a teendők között.

Amennyiben az előző két fejezetben leírtakat mindenképp meg szeretnénk valósítani, könnyen szembesülhetünk azzal, hogy a versenyzők számítógépeinek összeszerelésére és telepítésére szánt idő több, mint amennyi a verseny kezdetéig még rendelkezésünkre áll, *ezért minden esetben fogjunk bele időben az összeszerelésbe és telepítésbe.*

Az évek során – néha a szűkös emberi erőforrás, néha pedig a szűkös idő miatt – összegyűjtöttem számos hasznos technikát, technológiát, trükköt, amelyekkel a telepítés által felemésztett idő lényegesen redukálható.

1. *Ismerjük a hardvert!* Bár elsőre triviális gondolat, miszerint, úgy rendezzünk számítógépes programozási versenyt, hogy ismerjük az alkalmazott hardvert kívülről-belülről, a legtöbb esetben ez nem teljesül – minden igényt kielégítő mértékben.

   A legtöbb esetben ugyanis, a szervezőknek nem áll közvetlenül rendelkezésére a szükséges mennyiségű számítógép, ezért egyetemektől, iskoláktól vagy egyéb más intézményektől „bérlik" azokat. Ez viszont szükségképpen azt jelenti, hogy a szervezőknek igénybe kell vennie a számítógépek rendszergazdáinak segítségét, hiszen ők azok, akik ismerik a rendszer minden részletét.[3] A tapasztalat azt mutatja, hogy ezeket a rendszergazda–technikai szervező párbeszédeket már több, mint egy hónappal a verseny előtt érdemes megkezdeni.

   A párbeszédek során érdemes tisztázni a számítógépek – vagy típusok – főbb jellemzőit, különös tekintettel azokra a tulajdonságokra, amelyekről már szó volt korábban, a versenyt közvetlenül megelőző telepítés, az esetleges helyreállítás és a verseny után végrehajtandó visszaállítás lépéseit.

   Ezekre azért van szükség, mert az igénybe vett számítógépek legtöbbször nem számítógépes versenyek céljából vannak fenntartva, hanem egészen más feltételeknek, igényeknek, elvárásoknak kell megfelelniük – így a verseny feltételeihez drasztikus változtatások szükségesek, amelyekre előre fel kell készülni.

2. *Ismerjük a rendszert!* Az egyes számítógépeken kívül a rendszer egészéről is egységes és átfogó képpel kell rendelkezzünk, beleértve a számítógépes hálózatot, azok külső és belső kapcsolódási pontjait, a vonalak sebességét, terhelhetőségét, védelmi vonalait (tűzfalak, stb.), illetve a hálózat logikai működéséért felelős szervert vagy szervereket.

   Érdemes ügyelnünk az elektromos hálózatra is, a termek áramellátását megszakító kapcsolókra és a hálózati eszközök áramellátására, ugyanis előfordulhat, hogy a rendkívüli használat miatt – például egy hétvégén megrendezett verseny esetében – egyszerűen nincsenek áram alatt azok a hálózati eszközök, amelyek működését evidensnek vesszük.

3. *Telepítsünk előre, telepítsünk egyszer!* A számítógépek és rendszerek tulajdonságainak ismeretében *elméletileg* már akár otthon, vagy a versenyen résztvevő számítógépektől távol is felkészíthetünk egy teljes versenyzői konfigurációt az éles küzdelemre, amivel a külső rendszergazda idejét, illetve a saját, idegen számítógépes rendszerben eltöltött időnket tudjuk lényegesen redukálni.

   A *gyakorlatban* ez modern virtualizációs technikák alkalmazását (pl. QEMU, Virtual PC, VMware Workstation, stb.)[17] vagy nagy szaktudást igénylő „rendszer

[3] Manapság már előfordul, hogy az iskolában sem az informatika tanár látja el a rendszergazda feladatokat, ezért ez már az iskolákra is igaz, a versenyt szervező tanár igénybe veszi a rendszergazda tudását és idejét, a verseny (vagy érettségi, stb.) zökkenőmentes lebonyolításához.

a rendszerben" telepítést igényel, amely során egy már működő operációs rendszerbe, egy másikat telepítenek, amelyek akár önállóan is képesek működni.

4. *Telepítsünk egyszerre!* Ez látszólag az előzőnek ellentmondó javaslat lehet, azonban míg az előbbi a versenyzői konfigurációk funkcionalitásának összeállítását tűzi ki célul, az *egyszerre telepítés* a már kész szoftver-csomagok (*image-ek*; ami itt az operációs rendszer és a szoftverek együttesének egy kész csomagját jelenti) számítógépekre másolását takarja.

   A látszólag egyszerű lépés is komoly fejtörést okozhat, ugyanis sok számítógép és nagyméretű *image*-ek esetén számos fizikai korlátba ütközhetünk:

   - Amennyiben a programok és az operációs rendszer együttesen több helyet foglalnak el – akár tömörítve is, – mint egy CD vagy DVD kapacitása,
   - vagy nincs megfelelő olvasó a gépekben,
   - illetve, ha a számítógépek száma annyira nagy, hogy emberi vagy anyagi erőforrások hiánya miatt nem megoldható a lemezes telepítés, csak a hálózati telepítés marad, mint lehetséges alternatíva.
   - Nagy adatforgalom esetén a hálózat is korlátot jelenthet, hiszen ha minden számítógépre egyesével – akár egymás után vagy párhuzamosan – másoljuk fel a kész programokat, a számítógépek számával egyenesen arányosan nő a másolásra fordított idő – ami a tapasztalatok alapján órákat, akár napokat jelenthet.[4]

   A problémával szerencsére már mások is szembesültek, ezért egy kész technológiát érdemes segítségül hívnunk, a *multicaston alapuló szoftver elosztást*. A módszer lényege az, hogy egy számítógépről, egy olvasás során, egy hálózati kimenő forgalom alatt, elméletileg egyszerre telepíthetjük az összes számítógépet, amennyiben minden számítógépre ugyanazt az operációs rendszert és alkalmazásokat kell telepíteni.

   A gyakorlatban ez természetesen a hibakezelés és az egyéb járulékos hálózati forgalom miatt inkább kétszeres időt jelent, azonban az eredeti, számítógépek számával egyenesen arányos telepítési idő, így konstans, az egy számítógép telepítéséhez szükséges idő kétszeresére csökkenthető – pl. a Linuxos *udpcast*, vagy a közismert *Norton Ghost* programok segítségével.

5. *Módosítsunk egyszerre!* Az telepítéshez hasonlóan egy esetleges módosítás is rendkívül idő- és emberigényes feladat, hiszen minden számítógépen ugyanazokat a módosításokat kell egyszerre (vagy egymás után, de egyik gépet sem kihagyva!) elvégezni.

   Hatékony megoldása a problémának, Linuxos környezet esetén, a távoli parancssoros bejelentkezést biztosító szolgáltatás (*OpenSSH*), illetve az ezt használó szinkronizációs másolást támogató program telepítése a versenyzői gépekre (*rsync*). Ezzel megoldható a szükséges állományok, programok, versenyzői fájlok másolása,

[4] Tapasztalatok alapján 50 darab számítógép, 4GB-os *image*-ek, és garantált sebességű 100MBit-es full-duplex hálózat esetén a teljes telepítés körülbelül 5 órát vesz igénybe, ami a technikai paramétereket figyelembe véve, a közelmúltban, inkább átlagon felülinek mondható, mint átlagosnak.

illetve egyszerűbb szkriptek segítségével olyan feladatokat is könnyen és egyszerűen elvégezhetünk, mint például a verseny befejezését követő automatikus – és minden gépet egyformán érintő – adatmentés.

Egy hátránya van a távoli belépést alkalmazó módszernek, méghozzá az, hogy minden bejelentkezéskor – tehát minden kiadott parancs esetén annyiszor, ahány számítógépen az adott parancsot párhuzamosan végrehajtjuk – titkos jelszót kell megadnunk a számítógépnek. Fontos azonban az, hogy saját kényelmünket nem helyezhetjük a rendszer biztonsága elé, ezért az üres jelszó helyett érdemes az *OpenSSH* privát és publikus kulcsokat alkalmazó beléptetési módszereit áttekinteni. A kulcsok használatával elkerülhető, hogy a versenyzők egymás számítógépeit, azok adminisztrációra használt kommunikációs csatornáin keresztül, megtámadják és feltörjék.[5]

6. *Legyen tartalék* – azaz mind hardverből, mind szoftverből legyen kéznél előre telepített, azonnal versenybe állítható tartalék! A tapasztalat azt mutatja, hogy erről sajnos az esetek többségében elfeledkeznek a szervezők.

A fent leírt technikák természetesen az informatikai élet más szegleteiben is alkalmazhatók – például homogén gépparkok esetén, ahol az eredeti töredékére csökkenthető a szoftverek karbantartási vagy újratelepítési ideje.

Több iskolában jelent nehézséget a tömeges informatika érettségi. Általában előző nap még tanítás van azokban a termekben, ahol másnap az előírásoknak megfelelő számítógépes rendszereken a diákok vizsgáznak. Sok esetben emiatt a tanárok, rendszergazdák éjszaka telepítenek, finomhangolják a számítógépeket, ami kiváltható lenne a fent leírtak alkalmazásával.

3.1.4. Szerver(ek) és kiértékelő gép(ek)

Ha a szervezők alkotják egy verseny csontvázát, a versenyzői számítógépek az izomzatot, akkor minden bizonnyal a központi szerver- és kiértékelő számítógépek a szív és a tüdő.

A korábban említett általános hardveres és szoftveres javaslatokon kívül, ez utóbbiakra számos olyan szabály vonatkozik, amelyeket legjobban a szervereket üzemeltető rendszergazdák ismernek, ezért – amennyiben a szervezők között nincs olyan, aki hivatását tekintve rendszergazda – érdemes szakember tanácsát, segítségét vagy munkáját igénybe venni.

Az évek során a következő, központi szerverek által biztosítható, szolgáltatásokra volt szükségem:

- *hálózati címek kiosztása és „fordítása”*– webes környezetek köré szerveződő versenyek esetén minden versenyzői számítógépnek szüksége van egy egyedi azonosítóra, amivel a hálózaton kommunikálhat. A hálózaton belül egyedi IP-címek kiosztását végző szolgáltatás a *DHCP-szolgáltatás*. Ezzel szoros kapcsolatban van a *DNS-szolgáltatás*, amelynek feladata az előbb említett IP-címek (számok sorozata; pl. 192.168.1.10) és a logikus számítógépnevek (pl. computer_001.acm) közti leképezés megvalósítása.

[5] Érdemes a versenyzői gépek között – az egyszerű felhasználók által nem módosítható – szoftveres tűzfallal védeni ezeket az adminisztrációs szolgáltatásokat – ebben a példában: tiltani kell a versenyzői gép–versenyzői gép közti 22-es portra való csatlakozást.

- *Tűzfal* – már korábban is felmerült a versenyzői gépek közti adatforgalom tiltása, azonban ha hálózatban egyidejűleg több verseny is megrendezésre kerül, például egy belső, zárt és egy külső, nyílt, internetes verseny esetén, a belül található gépeket el kell tudni zárni a külvilágtól úgy, hogy a verseny lebonyolításáért felelős szervert viszont el tudják érni a külső versenyzők.

  Mindezek mellett egy erre a feladatra dedikált tűzfal számítógép újabb védvonala lehet a rosszindulatú versenyzők ellen vívott küzdelmünknek.

- *Központi időszinkronizálás* – a verseny kezdetének és végének ideje minden versenyen fontos momentum. Az idő-szolgáltatás feladata a gépek órájának megfelelő szinkronban tartása.

- *Webszolgáltatás* – függetlenül attól, hogy milyen kiértékelő rendszert alkalmazunk, a versenyeken publikálandó adatok közlésének legegyszerűbb módja egy webes szolgáltatás indítása. A webes tartalom előnye, hogy gyakorlatilag bármilyen versenyzői hardver és szoftver konfiguráción egységesen jeleníthető meg, biztonságos és biztosítható vele a kétirányú kommunikáció – így egyszerűen megoldható a feladatokra adott megoldások beszedése is.

- *Központi beléptető szolgáltatás* – amellyel a teljes rendszer felhasználóit és azok jogosultságait központilag adminisztrálhatjuk és ellenőrizhetjük. Ennek a verseny kezdetét megelőző néhány percben van kritikus szerepe, amelyről a 4.3. fejezetben lesz szó.

  A professzionális megoldások közé tartozik a *single sign on* technológia alkalmazása, amellyel elérhető, hogy a versenyzőknek csak egyszer kelljen felhasználónevüket és jelszavukat begépelni, az első bejelentkezésnél. A rendszer a módszer alkalmazásával képes az összes, később felmerülő, azonosítással kapcsolatos feladatot levenni a versenyzők válláról – pl. nyomtatás, megoldások feltöltése, stb.

- *Tárhelyszolgáltatás* – a verseny során előfordulhat olyan hardverhiba, amely miatt egy versenyzői számítógépet cserélni kell. A gépen dolgozó versenyző(k) elveszett ideje mellett az elveszett munka is fontos, ami kivédhető távoli – redundáns – tárhelyszolgáltatással. A versenyzőktől „távol” tárolt állományok esetében azonban még jobban oda kell figyelni a biztonságra és a jogosultsággal kapcsolatos beállításokra.

- *Naplózás* – a versenyzők tevékenységeinek ellenőrzése – például csalások elkerülése érdekében – csak központilag megoldható feladat, aminek egyik része a számítógépek *monitorozása*, illetve a naplóállományok gyűjtése. Ez utóbbit valósíthatjuk meg egy központi naplógyűjtő szolgáltatással.

- *Nyomtatás* – mivel nyomtatóból lényegesen kevesebb van, mint bármilyen más perifériából a teljes rendszerben, elkerülhetetlen, hogy több versenyző egy azonos nyomtatót használjon. A közös nyomtató és az illetéktelen kezekbe kerülő programkódok és információk veszélye miatt, a szervezőkre hárul a nyomtatót elhagyó papírok szétosztása. Ennek elengedhetetlen feltétele a dokumentumok egyértelmű azonosítása.

Könnyen megoldható és mindenki kényelmét szolgálja, ha a nyomtatószoftver képes a nyomtatásra küldött állományokat szintaxis-kiemeléssel együtt nyomtatni. Linuxos nyomtatószoftver alkalmazása esetén az *a2ps* programcsomag szép eredményeket produkál.

Ezek a hálózati- és rendszerszolgáltatások nem mind létfontosságúak, azonban mind a teljes rendszer adminisztrálásában, mind az esetlegesen felbukkanó hibák keresésében és a verseny gördülékeny lebonyolításában sokat segítenek.

Amik viszont mindenképp szükségesek egy modern programozási versenyen:

- *feladat publikáló és megoldás gyűjtő szolgáltatás* – például egy webes keretrendszer formájában, illetve
- *kiértékelő szolgáltatás* – lehetőleg több ugyanolyan számítógépen, úgy, hogy a kiértékelésre kiválasztott gépeknek semmilyen más feladata ne legyen. A kiértékelés ugyanis csak akkor objektív és igazságos, ha a kiértékelendő programok identikus környezetben végzik feladatukat.

A két rendkívül összefonódó feladatra már több szoftver-család készült a világ több pontján (*SIO*[18], *OnlineJudge*[19], stb.), ezért megtehetjük, hogy a később ismertetett tulajdonságok, elvárások és ajánlások alapján választjuk ki az értékelést végző programokat.

3.1.5. Feladatok

Az előkészületek eddig elvégzett feladatai mit sem érnek, ha a mérkőzés tárgya, a feladatok nincsenek körültekintően összeállítva. Minden versenyen alapvetően jól körülírható célok szabják meg a kiadott feladatokat, ezért a feladattípusokra, illetve a feladatokkal kapcsolatos pedagógiai döntésekre nem térek ki.

Azonban a verseny sikeressége szempontjából érdemes odafigyelni a feladatok összeállítása során a következőkre:

1. *Nehézség*. Már a versenytípusok ismertetése során is felmerült, hogy a tapasztalat szerint egy programozási verseny akkor sikeres, ha a versenyzők jól érzik magukat a versenyzés során. Ennek egyik feltétele, hogy a feladott példák, nehézségüket tekintve, lehetőleg a versenyzői és a szervezői oldal igényeit is kielégítsék, azaz
    - senki ne menjen haza megoldott példa nélkül,
    - ne legyen olyan példa amit senki nem old meg,
    - ne legyen olyan versenyző – vagy kevés ilyen legyen – aki minden példát megold és a verseny nagy részét unatkozással tölti el,
    - lehetőleg jól szórja a mezőnyt a feladatsor, azaz a verseny céljainak megfelelő szórású legyen az eredmény – például egy selejtező esetén a versenyzők két (vagy több) csoportra osztása, egy finálé során pedig a legjobbak elkülönítése a cél,
    - ne legyen túl sok „favágó” feladat.

    Érdemes szem előtt tartani a feladatok nehézségének meghatározásakor, hogy a programozási versenyeken elsősorban a „gondolkodás sebességét” mérjük, és csak

másodsorban a gépelési sebességet. Khera, Astrachan és Kotz, illetve a nyolcvanas évekbeli ACM versenyek feladatainak készítői szerint azok a jó feladatok, amelyek megoldása legfeljebb 200 sorból – vagy 150 függvényhívásból – megoldhatók.[20]

2. *Ellenőrzés* – a*milyen az értékelés, olyan a verseny*. Biztosak lehetünk abban, hogy ha a feladatok értékelésébe vagy az eredmények meghatározásába hiba csúszik, a verseny kudarcba fullad.

   Mivel számos feladat és feladattípus ellenőrzési stratégiája épít az automatizált kiértékelésre, a verseny előkészületei során mind a megoldásokat kiértékelő környezetet, mind a feladatmegoldásokat ellenőrző programokat alapos minőségbiztosítási folyamatokon kell végigvinni.

   Az évek során a következő, kiértékelést alapvetően érintő, problémákkal találkoztam:

   - *Hardverhiba* – a nyilvánvaló hardverhibákon kívül – rossz merevlemez, stb. – vannak olyanok, amelyek egy számítógép normál működése során nem derülnek ki, mert az operációs rendszer javítja a hibákat vagy egyszerűen nem veszik észre őket a felhasználók. Ilyen hardverhiba lehet például egy memóriamodul hibája, amely a mindennapi használat során csak elvétve okoz problémát – és amit más komponensek számlájára írunk. Egy programozási verseny esetében azonban minden bit számít, így elvárható, hogy a kiértékelést végző számítógépek kifogástalan állapotban legyenek.

     Ezt a hardver előzetes tesztelésével – memóriatesztek (pl. *memtest86*), merevlemez tesztek (pl. a linuxos *smartmontools*), stb. – biztosíthatjuk.

   - *Különböző architektúrák és függvénykönyvtárak* – interaktív feladattípus vagy olyan feladatok esetén, ahol a versenyző programjába a szervezők által készített külső *modult* beépítését írja elő a feladat, ügyelnünk kell arra, hogy azok a versenyzői számítógépeken és a ellenőrzést végző környezetben is jól működjenek.

     Végzetes hibát okozhat, ha a modult rossz processzortípusra fordítjuk vagy elfelejtjük beleilleszteni a külső függvényhívásoknak megfelelő könyvtárakat. Jó megoldás lehet egy egységes processzorcsalád előzetes kiválasztása – pl. i386, ami gyakorlatilag majdnem minden, itthon elterjedt, számítógéptípussal kompatibilis, – és a modulok statikus fordítása, ami feloldja a külső függvénykönyvtárak létének szükségességét.

   - *Tesztadatok hibája* – különböző operációs rendszereken, az alapokat biztosító függvénykönyvtárak eltérései miatt, a szöveges állományok kódolása, illetve a soremelést jelző karakter vagy karaktersor is eltérő lehet. Mivel a feladatmegoldások ellenőrzése során alapvetően szöveges állományokkal dolgoznak a programok, érdemes az ezzel kapcsolatos követelményeket és az elvárt formátumot előre tisztázni – például, ahogy az *UTF-8* kódolás defacto szabvánnyá vált az elmúlt pár évben[21] – a *kocsivissza és soremelés* karakterek alkalmazása jó megoldást jelenthet a különböző alkalmazások különböző értelmezéseire.

Az előre tisztázott formátumon túl, a programozási versenyek céljainak megfelelően, olyan tesztadatok előállítására törekedjünk, amelyek bármilyen programozási nyelv segítségével könnyen feldolgozhatók – kivéve ha a feladat részét képezi a bonyolult formátumú adatsor visszafejtése. Egyszerű tesztadatok esetén a szervezői tévedés is könnyebben kizárható, illetve a versenyzők sem vesznek el az adatok értelmezésében. Így a verseny nem a karakterfelismerésről és beolvasásról szól majd.

A helytelen mintamegoldások által generált, vagy más módon előállított, rossz tesztadatok ellen azonban nincs univerzális megoldás. A versenyfeladatok és tesztadataik előkészítése ezért rendkívüli figyelmet és szakértelmet igényel.

4. Verseny

„Az a jó, ha Bence alszik a verseny alatt."

– Benedek Balázs, 2006, ACM CERC főszervező

Ideális esetben a verseny ideje alatt, egy hozzám hasonló, informatikai infrastruktúráért felelős szervezőknek nincs dolga – legfeljebb a nyomtatóba kell egy újabb adag papírt beletenni. Ezt jelzi a rövid mottó is, miszerint a verseny főszervezője mindaddig nyugodt maradhat, amíg a rendszergazdák és a technikai szervezők csak figyelik a verseny menetét és nem sürögnek--pörögnek-forognak.

Ahhoz, hogy minden ilyen ideális rendben menjen, az előkészületek részeként, a már említett teendőkön túl, még számos feladat vár ránk.

4.1. Verseny előtt

Mind a több napos, mind a rövid, akár csak néhány órát igénybe vevő versenyek esetén a szabályok tisztázása elengedhetetlen része a versenynek. Bár minden programozási verseny egyedi a maga nemében, számítógépes versenyek esetében a következőket mindenképp ismertetnünk kell a versenyzőkkel:

- verseny *programja*, *időbeosztása*, *helyszínei*,
- *verseny típusának megfelelő szabályok* – pl. open-book, csapatverseny, limitált segítséggel, stb.
- *számítógépek használatával kapcsolatos tudnivalók*
    - operációs rendszer,
    - telepített programok,
    - belépéssel és kilépéssel kapcsolatos információk,
    - a felhasználói tárhellyel és az archiválással kapcsolatos tudnivalók – garanciák, hiba esetén használatos forgatókönyvek,
- kiértékelő rendszerrel kapcsolatos információk;

- belépéssel kapcsolatos információk,
- feladatok, publikus dokumentációk elérhetőségei,
- megoldások beküldésének, feltöltésének módja,
- eredmények elérhetőségei,
- az engedélyezett kommunikációs csatornák használatával kapcsolatos tudnivalók – pl. zsűri elektronikus elérhetőségei.

Az ismertetésnek több módja van, azonban írásos formában minden esetben le kell fektetnünk a verseny szabályait – az esetleges óvások és szabályok félreértelmezése végett.

Hosszabb verseny megrendezésekor, vagy ha módunk van rá, lehetőleg minden verseny előtt, szóban is fektessük le ezeket a szabályokat, így lehetőségük van a versenyzőknek visszajelzéseket adni azokkal kapcsolatban – illetve biztosak lehetünk benne, hogy legalább egyszer eljutott hozzájuk a megfelelő mennyiségű információ.

Előnyös megoldás az élő szó alkalmazása a számítógépes rendszer bemutatása során is, ahol a kiértékelő rendszer használatától kezdve, a számítógépeken lévő alkalmazásokon át, a bejelentkezéssel kapcsolatos összes tudnivalóra felhívhatjuk a versenyzők figyelmét.

Érdemes a versenyen már az első számítógépes időtöltés előtt kisorsolni és beosztani a versenyzőket az egyes termekbe. Egyrészt így a versenyzőknek van idejük megtalálni számítógépeiket, másrészt adminisztrációs és szervezési szempontból is egyszerűbb a versenyzők eligazítása. Amennyiben időben gondolunk a beosztásra, országok és nyelvek szerint is szétoszthatjuk a versenyzőket, amivel megnehezíthetjük a tiltott kommunikációt és a csalásokat.

4.2. Bemelegítő verseny

Szervezői és versenyzői oldalról is előnyös minden verseny előtt megismertetni a résztvevőkkel a helyszínt és az eszközöket. Sportok esetében ezt pályabejárásnak, programozási versenyeken bemelegítő versenynek nevezik.

Egyrészt a versenyzők gyakorlatban is megtapasztalhatják azt, amit a bemutatók során és a leírásokból megtudhattak a versenyről, másrészt a szervezők, a nagyszámú versenyző segítségével felderíthetik és javíthatják azokat a problémákat, amelyek csak éles használat során merülnek fel.

Fontos, hogy a bemelegítő verseny során minél több, az éles versenyben is előforduló folyamatot szimuláljunk, ezért érdemes az alábbiakat figyelembe venni:

1. ne legyen *hosszabb*, mint az éles verseny hosszának egynegyede, ami elég rövid ahhoz, hogy ne váljon unalmassá a bemelegítő, de biztosít elegendő időt arra, hogy a versenyzők beletanuljanak, megszokják a környezetüket,
2. körültekintően válasszuk ki a bemelegítő verseny során *feladott feladatokat,* ugyanis túl nehéz vagy túl sok feladat esetén a versenyzők a számítógépes környezetük felderítése helyett a feladatmegoldásra fognak koncentrálni,
3. a bemelegítő verseny alatt minden értelemben tartsuk magunkat az *éles verseny szabályaihoz*, ez segít megszokni a résztvevőknek a versenyt,

4. „ne essünk pánikba!”[42] – a felmerülő problémákat és hibákat ne kezdjük azonnal megoldani, gyűjtsük össze őket és az éles verseny idejéig fennmaradó időben, lehetőség szerint az összeset, fontosságuk szerinti sorrendben javítsuk,

5. a bemelegítő verseny után a versenyzői számítógépeken *állítsunk vissza mindent az eredeti állapotába*, azaz töröljük az összes versenyzői módosítást, környezeti beállítást és begépelt példaprogramot. Ennek legegyszerűbb módja a versenyzői számítógépek újratelepítése – például a korábban említett módszerek segítségével.

   Az eredeti állapot visszaállítását az egyenlő feltételek biztosítása teszi szükségessé, így minden versenyző azonos számítógépes konfigurációval, azonos környezettel kezdhet bele a versenybe.

4.3. Éles verseny

A bemelegítés folyamatait követően, a verseny következő mérföldköve az éles verseny lebonyolítása. Az alaphelyzetbe visszaállított és hibáktól mentes rendszer elindításán kívül két fontos feladata van a technikáért felelős szervezőknek a verseny elindítása előtt:

1. *Feladatlapok nyomtatása* – bár alapvetően elektronikus rendszerekkel segített versenyről van szó, a kézzel fogható, gyűrhető, firkálható papír szükséges eleme minden programozási versenynek.

   Salenieks és szerzőtársai[16] szerint egy átlagos programozási versenyen, amelyen 5 versenyző dolgozik egy csapatban 2 nyomtatott feladatsor segíti leginkább a hatékony csapatmunkát, ugyanis nem válnak szét egyénekre a feladatmegoldás során a korlátozott erőforrások miatt.

   Az elmúlt évek ACM versenyein szerzett tapasztalataim ezzel szemben azt mutatják, hogy mind az éles, mind a bemelegítő versenyen a 3 fős csapatok legalább 3 példányt kértek a feladatsorokból. Ennek oka a versenyzők felkészülési stratégiájában keresendő, ugyanis ACM versenyeken legtöbbször Informatikai Olimpiát megjárt versenyzők vesznek részt eredményesen, akik a verseny szabályaihoz igazodva, de legtöbbször lényegében egyénként oldják meg a feladatokat.

   Amennyiben a papír takarékos felhasználása fontosabb szempont, nyomtassuk a különböző feladatokat külön lapokra, így a szétbontható feladatsorral egyszerre többen is dolgozhatnak.

   A versenyzők maximális kényelmét a két nyomtatási módszer együttes alkalmazása biztosítja, azaz amikor minden versenyzőre jut legalább egy teljes, feladatonként legalább egy lapot tartalmazó feladatsor. Amennyiben a különböző nyelven beszélő versenyzők igényeit is ki szeretnénk elégíteni, anyanyelven írt feladatsort is mellékelhetünk a közös nyelven megírt mellett – ami elsősorban a fiatalabb korosztályok versenyein jellemző.

2. *Bejelentkezés tiltása a verseny kezdetéig* – amire azért van szükség, hogy a verseny kezdete előtt, az elhelyezkedő versenyzők ne kezdhessenek bele olyan

[42]… és legyen mindig nálunk törülköző!

előkészületekbe, amelyekkel behozhatatlan előnyre tesznek szert ellenfeleikkel szemben.

Érdemes ezt, a szabályok ismertetésén és a személyes felügyeleten kívül, szoftveresen, egy központi beléptető szolgáltatással támogatni, amely csak a verseny megkezdése után engedi belépni a gépek felhasználóit.

Ahogy már korábban jeleztem, a verseny elindulását követően alapvetően megváltozik minden, a versenyzők dolgoznak teljes erőbedobással, és ideális esetben, a szervezők fújhatják ki magukat.

A verseny során, annak típusának és szabályainak megfelelően, számos technikai részletre kell odafigyelni, amelyek szempontokként merülnek fel a segítséget nyújtó kiértékelő rendszer kiválasztásakor is. A helyes döntést csak megfelelő kérdések és válaszok segítségével hozhatjuk meg, amelyekhez, a verseny menetével kapcsolatosan, a következőket érdemes áttanulmányozni:

- *Támogatott programozási nyelvek*. Minél több, annál jobb. Egy támogatott vagy támogatandó nyelv esetében érdemes megfontolni, hogy milyen a versenyzők „kereslete" az adott nyelv iránt, halott vagy kifutó nyelvek támogatása nyilván indokolatlan.

  Érdemes elvetni azokat a nyelveket, amelyek alapvetően különböznek a korábbi versenyeken megszokottaktól[6], illetve azokat is, amelyek használatával egy versenyfeladat lényegesen egyszerűbben, könnyebben megoldható – pl. a legújabb magas szintű nyelveken lényegesen egyszerűbb egy listát kezelni, mint *C* vagy *Pascal* nyelveken.

- *Kiértékelés* . Mind a verseny szabályai, mind az informatika által biztosított lehetőségek erős korlátot határoznak meg a beadható megoldásokkal szemben, ezért az ellenőrzés kérdése talán a legösszetettebb egy programozási verseny során.

  Elsőként a megoldások által használt adatok bemenő és kimenő útját érdemes meghatároznunk. Régebben az adatállományokból olvasás és írás volt az elfogadott, az új rendszerek viszont már támogatják a *standard inputon* és *outputon* keresztül történő kommunikációt. Ennek előnye, hogy a feladatmegoldásba nem kell explicit módon belekódolni az olvasott és írt fájlok neveit, hanem a program egyszerűen a bemenetről olvashat és a kimenetre írhatja a megoldásokat. Ez a módszer a potenciális hibalehetőségeken kívül a versennyel kapcsolatos adminisztrációs teendőket is csökkenti.

  Második lépésben a forráskódból előállított futtatható program előállításával kapcsolatosan kell döntést hoznunk. Nagy versenyek esetén, ahol sok versenyző és sok feladat van, mindenképp szükséges a programok fordításának és futtatásának szeparálása. Így megoldható, hogy egy számítógép a fordítással foglalkozzon, a többi kiértékelésben segédkező „zsűri" pedig a tesztesetek futtatásával.

  A harmadik lépésben az ellenőrző tesztek és a futásukra szánt időmennyiségek alapján a kiértékelés párhuzamosításának szükségességéről kell döntenünk. Nagyméretű tesztesetek esetén azok bontásával lényegesen jobb reakcióidőt és

[6] ACM esetében valószínűleg sosem fog előfordulni logikai programozást megvalósító nyelv – pl. Prolog.

erőforrás felhasználást érhetünk el, ami a versenyzők feladatmegoldásaival szemben semmilyen további korlátot nem emel.

- Egybefüggő tesztesetek esetén záró karaktersorral jelezhetjük a tesztsorozat végét – például „0” számmal *(egykörös tesztelés – nem párhuzamosítható),*
- vagy megadhatjuk előre a tesztek számát *(egykörös tesztelés – nem párhuzamosítható)*, illetve
- külön állományokra darabolt tesztesetek esetén pedig egyszerűen az állomány vége jellel jelezhetjük a vizsgálat végét *(többkörös tesztelés – párhuzamosítható)*.

Pedagógiai és versenyzői szempontból mindegyiknek van előnye és hátránya, például egykörös tesztelés esetén ügyelni kell a memória helyes felszabadítására és az újrafoglalásra, azonban a bemenet és kimenet kezelése szempontjából nem számottevő az eltérés.

- *Feltöltések maximális száma.* A megoldások feltöltésének maximális számát egyrészt a felhasználható erőforrások ismeretében, vagy a verseny szabályai miatt érdemes korlátozni. Míg az előbbi eset az ellen nyújt védelmet, hogy a versenyzők ellenőrzés és tesztelés nélkül küldjenek be megoldásokat – felemésztve ezzel a kiértékelő rendszert, – az utóbbi didaktikai szempontból lehet előnyös olyan versenyek esetén, ahol az újból beküldött megoldások a versenyzők eredményeit nem befolyásolják negatívan.
- *Megoldások eredményeinek értékelése*. A kiértékelés második logikai lépése fizikailag könnyen elválasztható az elsőtől, ugyanis a *kiértékelő számítógépek* legtöbb esetben minden megoldást, minden szempontból kiértékelnek, és a következtetések levonását más programkomponensekre bízzák. Erre általában a párhuzamosítás és a rendszer hibatűrése miatt van szükség.

  A verseny elején ismertetett szabályok, a feladatokhoz rendelt limitek és a megoldások által adott válaszok összevetésével így egyértelműen meghatározható egy megoldás helyessége.

  A versenyt lebonyolító rendszer kiválasztásakor a mért programjellemzőket érdemes áttanulmányozni, ezek leggyakrabban a következők:

  - *Futási idő* – azaz a futtatható állomány tesztelése során felemésztett idő.
  - *Felhasznált memória* – amit a futtatható állomány a tesztelés során maximálisan elfoglalt – ezen belül a modern értékelő programok képesek a *heap* és *stack* értékeinek maximumát is megállapítani.
  - *Fordítási idő* – a fordítóprogram, fordítással eltöltött ideje. Számos versenyfeladatot, véges állapotterük miatt, már fordítási időben is meg lehet teljesen oldani, ami a futás során a bemenethez tartozó kereséssel teljesedik be.
  - *Forráskód maximális hossza* , illetve az előfordított állomány maximális hossza – amelyekre az előbbiek miatt van ugyancsak szükség,

- *Kilépési kód* – amelyre a programok szabályos működésének ellenőrizhetősége miatt van szükség. Például egy helytelenül befejeződő, memória felszabadítással nem foglalkozó program előállíthatja a helyes kimenetet a meghatározott időn belül, azonban az ilyen megoldásokat didaktikai szempontból érdemes elutasítani. A kilépési kód ellenőrzése ezt a célt szolgálja.
- *Egzotikus paraméterek* , mint a kommentek száma, a forráskód külalakja, hibakezelés mértéke, változónevek beszédessége, üres sorok száma, mind jó mérőszámai egy versenynek. Sajnos azonban a legtöbb kiértékelő rendszer az itt felsoroltakból nem támogat egyet sem.[7]

- *Feladatok emberi zsűrizése*. Minden rendszerbe csúszhat hiba, amelyet csak úgy detektálhatunk a verseny közben, ha folyamatosan figyeljük és ellenőrizzük annak működését. A kézi ellenőrzés eszköze egy, a zsűri számítógépekhez hasonló környezet, illetve a kiértékelő rendszer adattároló funkciója, amelyből a versenyzők megoldásait kinyerhetjük.

  Hiba, téves tesztadatok, hibás mintamegoldás vagy ellenőrzőprogramok miatt a kiértékelő rendszernek a felderítésén kívül az újraértékelés lehetőségét is támogatnia kell. Ennek feltétele, hogy mind a tesztadatok, mind a megoldások egységes és teljes rendszerre kiterjedő módon hozzáférhetők a szervezők számára.

- *Üzenetküldés*. A verseny során engedélyezett kommunikáció elektronikusan is ellenőrizhető formája lehet egy hétköznapi levelező rendszer, vagy egy kifejezetten a verseny céljainak megfelelő üzenetküldő rendszer. Mielőtt belefognánk a telepítésbe vagy implementálásba, érdemes végigböngészni a kiválasztott értékelő rendszer funkcióit és tudását összehasonlítani az alábbi elvárásokkal:
  - *Kétirányú, felügyelt kommunikációt* biztosítson, azaz a versenyző és a szervezők között mindig, a versenyzők között viszont csak a szabályok által engedélyezett esetben biztosítson kommunikációs lehetőséget a rendszer.
  - Lehetőségünk legyen *publikus kérdésekre, publikus válaszokat* és *privát kérdésekre privát válaszokat* adni, illetve a verseny szabályainak megfelelően a kérdések és válaszok láthatóságát változtatni.
  - Képes legyen kezelni és továbbítani *az értékelő rendszer* által generált üzeneteket.

  A legutolsó szempont miatt érdemes a saját vagy kiértékelő rendszerbe integrált üzenetküldést választani, az egyszerű levelező rendszer helyett.

- *Nyomtatás*. A nyomtatott dokumentumok egyértelmű azonosítása érdekében a nyomtatás támogatása lényeges szempont a kiértékelő rendszer kiválasztásakor. Más, a kiértékelő rendszertől független megoldást kell keresni, amennyiben az nem biztosít nyomtatási funkciót.

[7] Kevés olyan számítógépes verseny van, ahol ezek a paraméterek egyáltalán megjelennek bármilyen formában az eredményben.

- *Eredmények publikálása*. Azonnali kiértékelést alkalmazó versenyek esetén az aktuális eredmények publikálása mind a versenyzők, mind a kísérő tanárok számára fontos feladata a szervezőknek.

  A versenyen fokozhatjuk az izgalmakat, ha az eredmények listáját a befejezés előtt egy előre meghatározott idővel, például 1 órával, befagyasztjuk. Bár ez egy egyszerűen megvalósítható automatizmus, kevés kiértékelő rendszer támogatja.[8]

A verseny megrendezése nemcsak technikai feladatokból áll, a versenyzők kényelmére, étkeztetésére, hosszabb versenyeken elszállásolására és kikapcsolódására is gondolni kell, azonban írásomban ezekre nem térek ki részletesen, egy témát kivéve: süti és üdítő fogyasztását a számítógépek rendszergazdái általában nem szokták megengedni a véletlen balesetek elkerülése érdekében. Egy programozási verseny azonban olyan kivétel lehet, ahol hozzáértő és számítógéphez szokott emberek végeznek komoly, zavarást és megszakítást nem tűrő munkát. Ezért ha módunk van rá, oldjuk fel ezt a tilalmat arra a pár órára, a versenyzők kényelme érdekében.

5. Utómunkálatok

A verseny végeztével annak feladatai még nem érnek véget, amíg a versenyzők kipihenik magukat, a szervezőknek a következőket kell minden igényt kielégítő módon elvégezniük:

1. *Óvások, verseny közben felmerült reklamációk és hibák orvoslása* – amely legtöbbször egyáltalán nem vagy csak részben automatizálható feladatokat takar. A versenyt követő feladatoknak ez olyan része, amelyre nem lehet előre felkészülni.
2. *Versenyzői adatok, feladatok, megoldások archiválása* – amelyekből egyrészt az utókor szervezői tanulhatnak, másrészt az írásban, később beadott óvások miatt lehet szükség.
3. *Eredmények, oklevelek nyomtatása*.
4. *Statisztikák, feladatok és megoldások publikálása* . A versennyel kapcsolatos statisztikák, a feladatok és a megoldások szöveges publikálása általában nem jelent gondot egy verseny megszervezése során, azonban a tesztadatokat legtöbbször a verseny szervezői, saját érdekükben, általában a reklamációk elkerülése végett, nem hozzák nyilvánosságra.

   Hasonlóan a tesztadatokhoz, a versenyzők által beküldött megoldások sem szoktak felkerülni a publikálandó adatok közé, ami elsősorban szerzői jogokkal kapcsolatos megfontolások eredménye.

6. Konklúzió

Az elmúlt években számos programozási versenyen vettem részt szervezőként, illetve kísérő tanárként . Reményeim szerint a bemutatottak segítenek mind a tapasztalt szervezőknek, mind a készületeket támogató rendszergazdáknak, illetve a szárnyaikat próbálgató fiatal informatika tanároknak is számítógépes programozási versenyeik szervezésében. Bízom abban, hogy a leírtak segítségével időt és energiát takarítanak meg a szervezők, illetve abban is, hogy sikerült

[8] Leginkább csak azok, amelyeknél a versenytípus szabályai előírják ezt a funkciót.

rávilágítanom olyan területekre, amelyek eddig fel nem fedezett buktatókat jelentenek egy programozási verseny során.

7. Hivatkozások

1. *Central-European Olympiad in Informatics – 2005 Sárospatak* (2005), http://ceoi.inf.elte.hu/ceoi2005/
2. *Association for Computing Machinery's International Collegiate Programming Contest – Central-European Regional Contest* (2006), http://icpc.cs.bme.hu/
3. *Logo Országos Számítástechnikai Tanulmányi Verseny* (2007), http://logo.inf.elte.hu/
4. *Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Verseny* (2007), http://nemes.inf.elte.hu/
5. *Central-European Olympiad in Informatics* (2007), http://ceoi.inf.elte.hu/
6. *International Olympiad in Informatics* (2007), http://www.ioinformatics.org/
7. *The ACM International Collegiate Programming Contest* (2008), http://icpc.baylor.edu/
8. *ICFP (International Conference on Functional Programming) Programming Contest*, (2008), http://www.icfpcontest.org/
9. *Project Hoshimi* (2008), http://www.project-hoshimi.com/
10. *Core Wars* (2008), http://www.corewars.org/
11. *Topcoder* (2008), http://www.topcoder.com/
12. *ImagineCup* (2008), http://imaginecup.com/
13. *<19 Szabadfogású Számítógép Verseny* (2008), http://verseny.c3.hu/2007/
14. Struble, G. 1991. *Experience hosting a high-school level programming contest*. SIGCSE Bull. 23, 2 (May. 1991), 36-38.
15. Conlon, M. P. 2005. *RockTest: a programming contest management system.* J. Comput. Small Coll. 20, 5 (May. 2005), 27-35.
16. Salenieks, P. and Naylor, J. 1988. *Professional skills assessment in programming competitions.* SIGCSE Bull. 20, 4 (Dec. 1988), 9-14.
17. *Virtualization* (2008), http://en.wikipedia.org/wiki/Virtualization
18. *System Internetowy Olimpiady* (2008), http://sio.mimuw.edu.pl/index.phtml
19. Universidad de Valladolid: *OnlineJudge* (2008), http://icpcres.ecs.baylor.edu/onlinejudge/
20. Khera, V., Astrachan, O., and Kotz, D. 1993. *The internet programming contest: a report and philosophy.* SIGCSE Bull. 25, 1 (Mar. 1993), 48-52.

21. *Universal Character Set: ISO10646* (2008), http://en.wikipedia.org/wiki/ISO/IEC_10646