Problémaosztályok és algoritmusok

Program optimizálás

Cél

- Megváltoztatni a programot, hogy hatékonyabban működjön:
  - Gyorsabb
  - Kevesebb memóriát használ
  - Jobban kihasználja a számítógép forrásait
  - Kisebb tárhely, kevesebb sávszélesség
  - Kisebb energiafogyasztás

Példa

- Adjuk össze a számokat 1-től N-ig

```
ossz=0;   N darab összehasonlítás
for(i=1;i<=N;i++)   N darab inkrementálás
        ossz=ossz+i;   N darab összeadás
printf("%d",ossz);
```

- Optimizált változat (Gauss képlete):

```
printf("%d",N*(N+1)/2);
```

3 művelet

Optimizálás

- Az optimizálásra nincs általános szabály
- Általában egyik tényező optimizálása a többiek rovására megy – ezért ki kell választani, hogy mit optimizálunk.
- Például:
  - egy mobil rendszerben fontos a minimális energiafogyasztás
  - egy játék esetében fontos hogy ne szakadozzon, de nem baj, ha több DVD-t foglal el.
  - egy internetes szolgáltatásnál a tárhely majdnem korlátlan, de századmásodpercek alatt kell keressen több milliárd adat között.
- Optimizált algoritmust nehezebb utólag módosítani.

Példa

- Cseréljük ki két változó értékét!

```
pot = x;
x = y;
y = pot;
```

+memória
- művelet

- Másik megoldás

```
x ^= y;
y ^= x;
```

- memória
+művelet

Optimizálási szintek

- **Tervezési szint**: optimális algoritmusok kiválasztása
- **Forráskód szint**: A lassú pontok megkeresése és optimizálása
- **Fordító szint**: Olyan fordító használata amely optimálisabb forráskódot eredményez
- **Gépi szint**: olyan program írása amely maximálisan kihasználja a hardware adottságait
- **Futási szint**: a program különbözőképpen működik a futási hardware adottságai szerint.

Hatékonyságelemző (profiler)

- Az erőforráskihasználás és a lassú pontok azonosítása