Alapvető algoritmusok

8. előadás

Dr. Pătcaș Csaba

BABEŞ-BOLYAI TUDOMÁNYEGYETEM
Matematika és Informatika Kar

Tartalom

Injektív függvények

Feladat

Generáljuk az összes $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ injektív függvényt.

- Egy függvény injektív, ha ugyanazt az értéket nem veszi fel többször.
  Példa: $n = 2, m = 3$
  1 2
  1 3
  2 1
  2 3
  3 1
  3 2
- Melyik feladattal ekvivalens a fenti?

Injektív függvények

Feladat

Generáljuk az összes $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ injektív függvényt.

- Egy függvény injektív, ha ugyanazt az értéket nem veszi fel többször.
  Példa: $n = 2, m = 3$
  1 2
  1 3
  2 1
  2 3
  3 1
  3 2
- Melyik feladattal ekvivalens a fenti?
  Válasz: Generáljuk m elem összes n-ed rendű variációját!
- Megfogalmazható-e a feladat sakktáblás analógiával?

Injektív függvények

Feladat

Generáljuk az összes $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ injektív függvényt.

- Egy függvény injektív, ha ugyanazt az értéket nem veszi fel többször.
  Példa: $n = 2, m = 3$
  1 2
  1 3
  2 1
  2 3
  3 1
  3 2
- Melyik feladattal ekvivalens a fenti?
  Válasz: Generáljuk m elem összes n-ed rendű variációját!
- Megfogalmazható-e a feladat sakktáblás analógiával? ($n \times m$ méretű táblára kell n bástyát elhelyezni)

Partíciók

Feladat

Generáljuk az n szám partícióit! Partíció alatt azt a felbontást értjük, amelynek során az n számot felírjuk pozitív számok összegeként. Két partíciót különbözőnek tekintünk, ha az előforduló értékek, vagy azok sorrendje különbözik egymástól.

Példa: $n = 4$

$4 = 1 + 1 + 1 + 1$
$4 = 1 + 1 + 2$
$4 = 1 + 2 + 1$
$4 = 1 + 3$
$4 = 2 + 1 + 1$
$4 = 2 + 2$
$4 = 3 + 1$
$4 = 4$

Partíciók

Elemzés

- Az eddig látott feladatoknál a végeredmények mindig az $M_1 \times M_2 \times \ldots \times M_n$ halmaz részhalmazai voltak, vagyis az elemek száma mindig n volt és a megállási feltételünk $k > n$ alakú.
- A visszalépéses keresést alkalmazhatjuk általánosabb esetekre is, amikor a végeredmény az $M_1 \times M_2 \times \ldots \times M_i, i \leq n$ halmaznak eleme, vagyis az elemek száma a belső feltételektől függ.
- Az előző példából láthattuk, hogy a partíciók feladatában is változó, hogy egy végeredmény hány elemből áll.

Partíciók

Megoldás

- Az eredmény kódolása egyértelmű, a partíció tagjait fogjuk tárolni a `v` tömbben.
- A belső feltétel: `v[1] + v[2] + ... = n`
- Folytatási feltétel: `v[k] < n - (v[1] + v[2] + ... + v[k - 1])`
- Paraméterek: `v` - eredménytömb, `k` - az aktuális tag sorszáma, `s` - a megmaradt érték amit fel kell bontanunk
- Megállási feltétel: `s = 0` egy lehetőség, de mi `v[k] = s`-et fogunk használni a pszeudokódban.
- Kezdeti hívás: `Partíció(v, k = 1, s = n)`
- Lokális változó: `j`, amely felveszi az összes lehetséges értékét `v[k]`-nak
- Rekurzív hívás: `Partíció(v, k + 1, s - j)`

Partíciók

Pszeudokód

```
ALGORITMUS Partíció(v, k, s)
  MINDEN j = 1, s végezd el:
    v[k] = j
    HA (j < s) akkor
      Partíció(v, k + 1, s - j)
    KÜLÖNBEN
      Kiír(v, k)
    VÉGE(Ha)
  VÉGE(Minden)
VÉGE(Algoritmus)
```

Mit kellene módosítani, ha két partíciót akkor tekintenénk különbözőnek, ha az előforduló értékek különböznének, a sorrend nem számítana?

Partíciók

Elemzés

- Az előző algoritmus által generált megoldásokat ábrázolhatjuk egy gyökeres fa segítségével, hasonlóképpen a korábban látottakhoz.
- A gyökér egy üres sorozatot ábrázol és az ezt következő szinteken a `k+1`. szintre vezető élek mellé a `v[k]` értékeket rendeljük.
- Minden végeredmény a fa egy ágának felel meg, amely összeköt egy levelet a gyökérrel.
- Láthatjuk, hogy az eddigi fákkal ellentétben, a levelek különböző szinteken helyezkednek el.
- A backtracking módszer tulajdonképpen egy mélységi bejárással építi fel ezt a fát.

Részhalmazok (másképp)

Emlékezzünk a részhalmazok feladatára:

Feladat

Generáljuk az $\{1, 2, \ldots, n\}$ halmaz összes részhalmazát!

- Korábban a Descartes-szorzat segítségével oldottuk meg a feladatot, de a tanult új módszerek segítségével más megoldásmód is lehetséges.
- Kódoljuk a `v` tömbben a részhalmaz elemeit.
- Ahhoz, hogy egy részhalmaz csak egyszer jelenjen meg, az elemeit növekvő sorrendben generáljuk.
- Ehhez, a `MINDEN` ciklust `v[k - 1] + 1`-től indítjuk, emiatt kezdetben `v[0]`-t 0-ra állítjuk.
- Ezzel a módszerrel lexikografikus sorrendben kapjuk meg a részhalmazokat.
- Minden új elem generálása új részhalmazhoz vezet.

Részhalmazok (másképp)

Pszeudokód

```
ALGORITMUS Részhalmazok(v, k, n)
  MINDEN j = v[k - 1] + 1, n végezd el:
    v[k] = j
    Kiír(v, k)
    Részhalmazok(v, k + 1, n)
  VÉGE(Minden)
VÉGE(Algoritmus)
```

Ismétlés

Belső és folytatási feltételek

- A belső feltételek írják le, hogy egy generált megoldást (eredményt) mikor tekintünk végeredménynek, vagyis ezek feladatfüggőek.
- A folytatási feltételek írják le, hogy mikor lépünk egy szintről a következőre, vagyis ezek implementációfüggőek.
- Például: Mik a belső feltételek és mik a folytatási feltételek a Descartes-szorzat feladatánál?

Ismétlés

Belső és folytatási feltételek

- A belső feltételek írják le, hogy egy generált megoldást (eredményt) mikor tekintünk végeredménynek, vagyis ezek feladatfüggőek.
- A folytatási feltételek írják le, hogy mikor lépünk egy szintről a következőre, vagyis ezek implementációfüggőek.
- Például: Mik a belső feltételek és mik a folytatási feltételek a Descartes-szorzat feladatánál?
- Válasz: Nincs egyik sem, mivel minden megoldást elfogadunk!
- Mi a helyzet a királynők feladatánál?

Ismétlés

Belső és folytatási feltételek

- A belső feltételek írják le, hogy egy generált megoldást (eredményt) mikor tekintünk végeredménynek, vagyis ezek feladatfüggőek.
- A folytatási feltételek írják le, hogy mikor lépünk egy szintről a következőre, vagyis ezek implementációfüggőek.
- Például: Mik a belső feltételek és mik a folytatási feltételek a Descartes-szorzat feladatánál?
- Válasz: Nincs egyik sem, mivel minden megoldást elfogadunk!
- Mi a helyzet a királynők feladatánál?
- Válasz: A belső feltételek, hogy úgy helyezzük el az összes királynőt, hogy ne üssék egymást. A folytatási feltételeket a `NemTámad` függvényben írtuk le a megoldásban.
- Gondolkodási téma: ugyanez a kérdés a *Részhalmazok (másképp)* feladat esetén.

Ismétlés

A visszalépéses keresésnek megfelelő fa

- A fa gyökere az üres megoldásnak felel meg és minden lentebb lépéssel egy-egy elemet hozzáadunk az aktuálisan generált megoldáshoz.
- Levélhez akkor érünk, amikor találtunk egy végeredményt, vagy amikor nem teljesültek a folytatási feltételek, vagyis az adott ágon már biztosan nem juthatunk végeredményhez.
- Speciális eset áll fenn a *Részhalmazok (másképp)* feladatnál. Miért?

Ismétlés

A visszalépéses keresésnek megfelelő fa

- A fa gyökere az üres megoldásnak felel meg és minden lentebb lépéssel egy-egy elemet hozzáadunk az aktuálisan generált megoldáshoz.
- Levélhez akkor érünk, amikor találtunk egy végeredményt, vagy amikor nem teljesültek a folytatási feltételek, vagyis az adott ágon már biztosan nem juthatunk végeredményhez.
- Speciális eset áll fenn a *Részhalmazok (másképp)* feladatnál. Miért?
- Itt nem csak levelek, hanem belső csúcsok is reprezentálhatnak végeredményt.

Pénzösszeg kifizetése

Feladat

Írjuk ki az S összeg minden lehetséges kifizetési módját $b_1, b_2, \ldots, b_n$ címletű bankjegyekkel! Minden bankjegyből elegendő mennyiség áll rendelkezésre.

- Az eredmény kódolása: a `darab` tömb `k`. eleme azt fogja tárolni, hogy a b_k címletű bankjegyből hány darabot használunk.
- A belső feltétel: $\sum_{k=1}^{n} \texttt{darab[k]} \cdot b_k = S$
- A folytatási feltétel: $\texttt{darab[k]} \cdot b_k < S - \sum_{i=1}^{k-1} \texttt{darab[i]} \cdot b_i$

Pénzösszeg kifizetése

Elemzés

- Paraméterek: `darab` - tömb, `k` - változó, `sum` - megmaradt összeg
- A folytatási feltétel alapján `darab[k]` $\cdot\, b_k <$ `sum`, ezt átrendezve kapjuk, hogy `darab[k]` $< [$`sum`$/b_k]$
- Megállási feltétel: ha `sum = 0` vagy `k` $= n$ és `sum` osztható b_n-el.
- Kezdeti hívás: `Fizet(darab, k = 1, sum = s)`
- Rekurzív hívás: `Fizet(darab, k + 1, sum - darab[k] * b[k])`

Pénzösszeg kifizetése

Pszeudokód

```
ALGORITMUS Fizet(darab, k, sum)
  HA (k = n) akkor
    HA ((sum % b[n]) = 0) akkor
      darab[n] = sum / b[n]
      Kiír(darab, n)
    VÉGE(Ha)
  KÜLÖNBEN
```

Pénzösszeg kifizetése

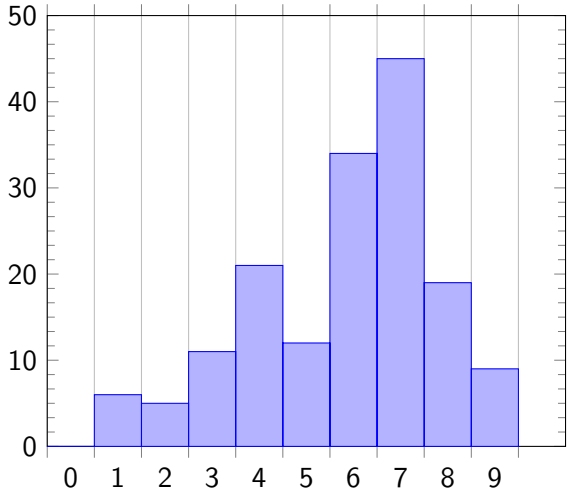
Pszeudokód

```
    MINDEN darab[k] = 0, sum / b[k] végezd el:
      maradt = sum - darab[k] * b[k]
      HA (maradt = 0)
        Kiír(darab, k)
      KÜLÖNBEN
        Fizet(darab, k + 1, maradt)
      VÉGE(Ha)
    VÉGE(Minden)
  VÉGE(Ha)
VÉGE(Algoritmus)
```

Tartalom

Írásbeli vizsga

Statisztika



- Átmenési arány: 73.4% (tavaly 51.2%)
- Átlag: 5.8 (tavaly 4.5)
- Medián: 6 (tavaly 5)
- Legnagyobb pontszámok: 28.14, 27.50, 26.75 (tavaly: 26.33, 25.92, 25.42)

Feladat

Írj rekurzív alprogramot pszeudokódban, amely megszámolja, hogy egy adott x természetes számnak, hány y-al egyenlő számjegye van!

- Hibás volt a előadásban szereplő algoritmus, $x = 0, y = 0$ esetre nem működött helyesen.
- Ezért nem vontunk le.
- Épp ideje volt, hogy valaki észrevegye ennyi év után, de Adrienne-nek már nem adhatok rá pluszpontot :)

Feladat

Írd le pár mondatban, hogy milyen feladatot old meg a *Kiválogatás helyben* programozási tétel!

- Keveri a szétválogatással
- Nem a *helyben* változatról ír
- Nem említi a memóriabonyolultságot (vagy hogy helyben dolgozik)
- Nem említi, hogy két változata van, aszerint, hogy a sorrend meg kell-e maradjon
- Stabilitásról jellemzően csak rendezések esetén beszélhetünk

Feladat

Mit nevezünk *stressztesztelésnek*, vagy *stress testing*-nek? Mikor érdemes alkalmazni és mik a potenciális hátrányai?

- Nem említi, hogy automatizált módszer
- Algoritmikai értelmezésben szükséges hozzá egy garantáltan helyes megoldás, aminek a kimenetével hasonlítunk.
- Nem feltétlenül „nagy” tesztesetekre alkalmazzuk, gyakrabban véletlen tesztekre.
- Akkor alkalmazzuk, amikor már az átlátszó doboz módszerével alaposan teszteltünk és nem találtunk megoldást, „végső megoldásként”.
- Hátránya, hogy könnyen ellustulhat a gondolkodásunk a gyakori alkalmazásával.
- Az esetek többségében nem időigényes.
- Nincs köze a programozó stressz-szintjéhez :)

Feladat

Melyik bonyolultsági osztály írja le a legpontosabban...

- Ha valaki felír több mint egy bonyolultsági osztályt (hátha közte van a jó is, majd válogat a tanár), az nem fog pontot érni.
- Aszíptotikus jelölés nélkül nem beszélünk bonyolultsági osztályról, esetleg csak növekedési rendről.
- A vizsgán megjelenő kérdésekben mindig Θ-val volt a legpontosabb leírható a válasz (de ez nem feltétlenül van mindig így...)
- Ha valak olyat ír, hogy ~~$O(n) = n$~~, vagy ~~$\Theta(n) = n$~~, az a jelölések helytelen használatát mutatja, hiszen nem függvényekről van szó, melyeknek változója az n, hanem végtelen halmazokról.

Feladat

Mit nevezünk *indentálásnak*?

- Sajátmagával definiálja (rekurzívan) a fogalmat: „az indentálás tördelés”
- Nem említi, hogy célja az olvashatóság
- Belekeveri az azonosítók elnevezését, alprogramokra osztást, kommenteket stb.

Feladat

Miért van szükség a könyvészet (irodalomjegyzék) megadására egy egyetemi előadás esetén?

- Mert hiteles forrás az internettel szemben
- A diáknak plusz magyarázatot és gyakorlófeladatokat nyújt
- A plagizálás elkerülése végett a tanárnak meg kell adnia

Feladat

Miért ellenjavasolt a `pow` függvény használata C/C++-ban?

- Mert pontossági problémák léphetnek fel a lebegőpontos valós számok használata miatt
- Mivel nem ismerjük a működését, nincs információnk a hatékonyságáról sem
- Nem feltétlenül váltható ki gyorshatványozással (csak ha a hatványkitevő egész szám)

Feladat

Milyen feladatot próbál megoldani az alábbi algoritmus? Mi benne a hiba?

- Orosz szorzás
- Rossz a megállási feltétel $b = 1$ helyett $a = 0$ kellene
- Elfogadható az az értelmezés is, hogy számrendszerek közötti konverziót próbál végezni
- A gyorshatványozás már túlzottan különbözik, ezért nem ér pontot

Írásbeli vizsga

Gyakori hibák

A legfontosabb kérdés: Hány éves Cecília és a testvérei?
9, 4, 4 (akkor is ha nem kolozsvári buszon ülnek a beszélgetéskor)
Miért nem jó a 16, 3, 3?

Jöhetnek a jegyek?

Még nem!

- Hiányzó pontszám, nem értem miért kaptam ennyi pontot → lásd javítótanárok
- ~~„Nem kaphatnék még egy fél pontot erre a feladatra?”~~

Javítótanárok:

- Rekurzív számjegyszámolás: Garfield Adrienne
- Radixsort: Lieb Hanna
- Aszimptotikus jelölések: Vekov Géza
- Eukleidész algoritmusa: Portik Ábel
- Többi: Pătcaș Csaba