Generare submulțimi

Fie n un număr natural nenul. Să se genereze toate submulțimile mulțimii {0, 1, …, n−1}.

Soluție

Vom reprezenta submulțimile prin vector caracteristic (un vector S cu n componente 0 sau 1, având semnificația S[i]=1, dacă i aparține submulțimii şi 0 în caz contrar). Problema se reduce astfel la generarea tuturor secvențelor binare de lungime n.

Pentru a rezolva această problemă vom utiliza un algoritm de tip succesor. Mai exact, un astfel de algoritm funcționează astfel:

Pas 1. Se inițializează soluția cu cea mai mică configurație posibilă (în cazul nostru, inițializăm vectorul S cu 0, ceea ce corespunde mulțimii vide);

Pas 2. Cât timp este posibil (există succesor) se execută:

- **se afişează configurația curentă**
- **se generează configurația următoare (în cazul nostru, vom considera că elementele vectorului reprezintă cifrele unui număr scris în baza 2, la care vom aduna 1).**

```
#include <fstream.h>
#define NMax 100
int main()
{int n, S[NMax], i, gata=0;
 ofstream fout("subm.out");
 cout<<"n= "; cin>>n;
 for (i=0; i<n; i++) S[i]=0; //initializare
 while (!gata)
     {//afisam configuratia curenta
      for (i=0;i<n;i++) if (S[i]) fout<<i<<' '; fout<<'\n';
      //generam configuratia urmatoare prin adunare binara
      for (i=0; i<n && S[i]; i++) S[i]=0;
      if (i==n) gata=1; /* este ultima configuratie  */
          else S[i]=1; }
 fout.close();
 return 0; }
```

Generare elemente produs cartezian

Fie n un număr natural (n>1) şi L=(l_1, l_2, ..., l_n) n numere naturale nenule. Să se determine în ordine lexicografică[1] toate elementele produsului cartezian {1, 2, ..., l_1}x{1, 2, ..., l_2}x...x{1, 2, ..., l_n}.

1. Fie x=(x_1, x_2, ..., x_n) şi y=(y_1, y_2, ..., y_m). Spunem că x precedă pe y din punct de vedere lexicografic dacă există k astfel încât x_i=y_i, pentru orice i<k şi x_k<y_k sau k>n.

De exemplu pentru n=3 şi L=(2, 3, 2), elementele produsului cartezian sunt: (1,1,1), (1,1,2), (1,2,1), (1,2,2), (1,3,1), (1,3,2), (2,1,1), (2,1,2), (2,2,1), (2,2,2), (2,3,1), (2,3,2). Observați că produsul cartezian are $l_1*l_2*...*l_n$ elemente.

Soluție

Vom reprezenta un element al produsului cartezian ca un vector E cu n elemente, unde $E[i] \in \{1, 2, ..., l_i\}$.

Pentru a genera toate elementele produsului cartezian în ordine lexicografică, vom aplica tot un algoritm de tip succesor:

Pas 1. Inițializăm vectorul E cu 1 (cel mai mic element al produsului cartezian, din punct de vedere lexicografic).

Pas 2. Cât timp este posibil (mai există succesor)

– afişăm elementul curent;

– generăm elementul următor; în acest scop căutăm prima componentă (începând din dreapta către stânga) care poate fi mărită (adică E[i]<L[i]); dacă găsim o astfel de componentă o mărim, şi repunem pe 1 toate componentele următoare; dacă nu găsim o astfel de componentă deducem că generarea s-a încheiat, acesta a fost cel mai mare element din punct de vedere lexicografic.

```
#include <fstream.h>
#define NMax 100
int main()
{int n, L[NMax], E[NMax], i, gata=0;
 ofstream fout("pc.out");
 //citire si initializare
 cout<<"n= "; cin>>n;
 for (i=0; i<n; i++) {cin>>L[i]; E[i]=1;}
 while (!gata)
      {//afisam configuratia curenta
       for (i=0; i<n; i++) fout<<E[i]<<' '; fout<<endl;
       //generam configuratia urmatoare
       for (i=n-1; i>=0 && E[i]==L[i]; i--) E[i]=1;
       if (i<0) /* acesta a fost ultima configuratie */
          gata=1;
          else E[i]++; }
 fout.close(); return 0;}
```

1. *Problema instructorului de schi*

Un instructor de schi are la dispoziție n perechi de schiuri ($n \leq 50$) pe care trebuie să le distribuie la n elevi începători. Scrieți un program care să distribuie cele n perechi de schiuri astfel încât suma diferențelor absolute dintre înălțimea elevului şi lungimea schiurilor atribuite să fie minimă.

2. *Mulțimi*

Se citesc de la tastatură două mulțimi `A` și `B` cu `n`, respectiv `m` elemente numere naturale mai mici decât `1000`. Să se determine intersecția și reuniunea celor două mulțimi. Implementați mai întâi o mulțime ca un vector în care memorați elementele mulțimii, apoi prin vector caracteristic. Realizați o comparație între eficiența implementării celor două operații cu cele două reprezentări.

31. *Numere*

Să se genereze toate șirurile formate din `n` cifre, fiecare șir generat având următoarele proprietăți:

– conține numai cifre din mulțimea {`1`, `2`, `3`, `4`};

– oricare două cifre alăturate sunt fie ambele pare, fie ambele impare.

Numărul natural `n` (`3≤n≤15`) se citește de la tastatură. Toate soluțiile vor fi scrise una după alta, cu spații între soluții, fiecare șir fiind scris fără spații între cifrele ce-l formează. De exemplu, pentru `n=3`, se afișează (nu neapărat în această ordine) șirurile: `111 113 131 133 311 313 331 333 222 224 242 244 422 424 442 444` (variantă bacalaureat august 2003)

32. *Schi*

Un concurs de schi se desfășoară astfel:

– fiecare cursă are două runde;

– în runda `1`, `N` schiori, numerotați de la `1` la `N`, iau startul în această ordine; când primul schior din prima runda (numerotat cu `1`) termină cursa avem timpul de care schiorul a avut nevoie pentru a parcurge traseul;

– pentru fiecare schior ce urmează dispunem de diferența de timp dintre timpul său și timpul schiorului care este în acel moment lider al cursei, deci cu cel mai bun timp;

– doar cei mai buni `M` schiori (ca timp) se califică pentru runda 2 și ei vor porni în cursă în ordinea descrescătoare a timpului obținut în runda `1` (ultimul schior calificat pentru runda 2 va porni primul, în timp ce primul clasat în runda `1` va porni ultimul);

– în runda 2 vom dispune inițial de timpul total al primului schior care pleacă în cursă (suma dintre timpul realizat în runda 1 și timpul realizat în runda 2), iar apoi pentru fiecare dintre următorii schiori vom avea diferența de timp dintre timpul său total și timpul total al schiorului cu cel mai bun timp până în acel moment.

Scrieți un program care va determina învingătorii după terminarea cursei. Presupunem că nu se poate întâmpla ca doi schiori să obțină același timp, nici după runda `1`, nici după terminarea cursei.

Date de intrare / ieșire

Prima linie a fișierului de intrare `schi.in` conține două valori întregi `N` și `M` separate printr-un singur spațiu, reprezentând numărul total de schiori, respectiv numărul de schiori calificați pentru runda `2`. Linia a doua conține timpul primului

schior din runda 1, apoi, următoarele N-1 linii conţin diferenţele de timp pentru ceilalţi N-1 schiori din runda 1. Următoarea linie conţine timpul primului schior după runda 2, apoi, pe următoarele M-1 linii diferenţele de timp pentru ceilalţi M-1 schiori după runda 2.

Prima linie a fişierului de ieşire schi.out va conţine numărul de ordine al schiorului clasat pe primul loc (aur), linia a doua numărul de ordine al schiorului clasat pe locul 2 (argint), iar linia a treia numărul de ordine al schiorului clasat pe locul 3 (bronz).

Restricţii

- 3 ≤ M ≤ N ≤ 100
- Timpul pentru orice cursă este între 10 şi 300 secunde.
- Timpii sunt numere reale cu cel mult două cifre zecimale.

Exemple

schi.in	**schi.out**	**schi.in**	**schi.out**
3 3 25.13 +1.14 +2.18 45.08 +2.14 +3.11	3 2 1	4 3 29.18 +2.18 +0.05 +1.13 54.22 +1.23 +1.11	4 1 3

Temă:

http://campion.edu.ro/arhiva - problemele minmax si submult