Aplicaţii la Principiul lui Dirichlet

Mirel Coşulschi
mirelc@central.ucv.ro

May, 2023

1 Probleme

1. Problema `https://www.nerdarena.ro/problema/secvrest`

   (SecvRest, `https://www.nerdarena.ro/problema/secvrest`)

   Rezolvare: Fie $v_1, v_2, \ldots, v_n$ elementele şirului de valori.

   Elementele şirului a se calculează astfel:

$$\begin{aligned} a_1 &= v_1 \\ a_2 &= v_1 + v_2 \\ &... \\ a_n &= v_1 + v_2 + ... + v_n \end{aligned}$$

   Pentru şirul de valori $a_1 \mod n, a_2 \mod n, \ldots, a_n \mod n$ avem una dintre următoarele situaţii:

   - cel puţin una dintre valori este 0, adică $\exists k \in \mathbb{N}$ astfel încât $a_k \mod n = 0$.
     Atunci soluţia problemei este perechea `1 k`.
   - nu avem nicio valoare nulă. Vom aplica *Principiul lui Dirichlet* pentru şirul de valori $a_1 \mod n, a_2 \mod n, \ldots, a_n \mod n$ şi mulţimea de valori $\{1, 2, \ldots, n-1\}$.
     Vor exista cel puţin două valori $i < j$ astfel încât $a_i \mod n = a_j \mod n$. Prin urmare se verifică relaţia $(v_{i+1} + v_{i+2} + ... + v_j) \mod n = 0$.
     Solutia în acest caz este perechea de valori `(i+1) j`.

   Nu este necesar să memorăm valorile şirului v_i, $i = \overline{1,n}$ deoarece vom păstra valorile sumelor $a_k = \sum_{i=1}^{n} v_i$.

   Listing 1: `secvrest.c`

```
#include <stdio.h>

#define NMAX 100001

int a[NMAX];
int f[NMAX];
```

```
int main() {
  FILE *fin, *fout;
  int n, i, v, k;

  fin = fopen("secvrest.in", "r");
  fout = fopen("secvrest.out", "w");

  fscanf(fin, "%d", &n);
  for (i = 1; i <= n; i++) {
    fscanf(fin, "%d", &v);

    // calculam suma valorilor primilor i termeni
    a[i] = a[i-1] + v;
  }

  i = 1;
  while (i <= n) {
    // restul impartirii sumei primelor i numere la n
    k = a[i] % n;

    // daca restul impartirii este 0 sau daca acest rest a mai fost
    // obtinut anterior
    if ((k == 0) || (f[k] > 0)) {
      fprintf(fout, "%d %d\n", (k > 0) ? f[k] + 1 : 1, i);

      i = n;
    } else {
        // pastram indicele elementului final al secventei 1 ... i
        f[k] = i;
    }

    i++;
  }

  fclose(fin);
  fclose(fout);

  return 0;
}
```

2. Problema `https://www.pbinfo.ro/probleme/1262/subsecv`

   (subsecv, `https://www.pbinfo.ro/probleme/1262/subsecv`)

   Rezolvare:

Listing 2: `subsecv.c`

```
#include <stdio.h>
#include <string.h>

#define NMAX 10001
```

```
int a[NMAX];
int f[NMAX];

int main() {
  FILE *fin, *fout;
  int n, i, left, right;
  long long s;

  fin = fopen("subsecv.in", "r");
  fout = fopen("subsecv.out", "w");

  fscanf(fin, "%d", &n);
  for (i = 1; i <= n; i++) {
    fscanf(fin, "%d", &a[i]);
  }

  left = n;
  for (i = 1, s = 0; i <= n; i++) {
    // totalul elementelor de la 1 la i
    s += a[i];

    // daca suma tuturor elementelor citite pana la momentul curent
    // este divizibila cu n
    if (s % n == 0) {
      // pastram limita din stanga a secventei curente
      left = 1;
      // si limita din dreapta a secventei curente
      right = i;
    } else {
        // daca secventa al carei rest la impartirea cu n are valoarea r
        // (s % n = r) este prima
        if (f[s % n] == 0) {
          // salvam in vectorul f limita din dreapta a secventei 1...i
          f[s % n] = i;
        } else {
            // daca am mai gasit o secventa al carei rest la impartirea
            // cu n sa fie r, se verifica daca secventa curenta nu are
            // indicele stang mai mic decat cea anterioara
            // (f[s % n] + 1 < left), sau daca exista mai multe solutii
            // cu indicele stanga minim daca secventa curenta nu are
            // indicele drept mai mic decat cea anterioara
            if ((f[s % n] + 1 < left)
                || ((f[s % n] + 1 == left) && (i < right))) {
              // pastram limita din stanga a secventei curente
              left = f[s % n] + 1;
              // pastram limita din dreapta a secventei curente
              right = i;
            }
        }
    }
  }
```

```
  fprintf(fout, "%d %d\n", left, right);

  fclose(fin);
  fclose(fout);

  return 0;
}
```

3. Problema https://www.pbinfo.ro/probleme/4432/struguri

(Struguri, ONI 2023, clasa a VIII-a, https://www.pbinfo.ro/probleme/4432/struguri)

Rezolvare:

Listing 3: struguri.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define NMAX 20000

typedef struct pair {
          int x;     // valoarea
          int idx;   // pozitia initiala in vector
        } PAIR;

typedef struct answer {
          int n;      // numarul de carute
          int carry;  // cantitatea transportata de o caruta
          int m;      // numarul de gramezi transportate
          int *t;     // indicii gramezilor transportate
        } ANSWER;

PAIR v[NMAX + 1];     // vectorul de valori
int first[NMAX + 1];  // first[r] - indicele unde apare prima data o suma
                      //            cu restul r
int last[NMAX + 1];   // last[r] - indicele unde apare ultima data o suma
                      //           cu restul r
int s[NMAX + 1];      // suma[k] - suma tuturor valorilor elementelor din
                      //           secventa 1...k
ANSWER a[NMAX];       // a[i] - valorile de afisat pentru tura i

/*
    Functia determina pentru un vector de valori cu n elemente, cea mai
  lunga subsecventa a.i. suma elementelor sa fie divizibila cu n.
  @param v - valorile sirului
  @param n - numarul de elemente al sirului v
  @param left - adresa ce pastreaza indicele din stanga al celei mai lungi
                subsecvente a.i. suma elementelor sa fie divizibila cu n.
  @param right - adresa ce pastreaza indicele din dreapta al celei mai lungi
                 subsecvente a.i. suma elementelor sa fie divizibila cu n.

    Se calculeaza in s[i] - suma valorilor primelor i elemente
```

```
  first[r] - indicele j unde apare prima data o suma a.i. s[j] % n = r
  last[r] - indicele j unde apare ultima data o suma a.i. s[j] % n = r

  - fie vom avea resturile de la 0 la (n-1) si atunci raspunsul este last[0]
  - fie nu vom avea nicio suma cu restul 0 si conform principiului lui
  Dirichlet vor exista cel putin doua sume, s[u] si s[v], u < v, astfel
  incat sa aiba resturi egale la impartirea cu n.
    Secventa va fi u+1, ..., v.
*/
void compute(PAIR* v, int n, int* left, int* right) {
  int i, r;

  memset(first, 0, sizeof(first));
  memset(last, 0, sizeof(last));

  for (i = 1; i <= n; i++) {
    s[i] = s[i - 1] + v[i].x;

    r = s[i] % n;
    if (first[r] == 0) {
      first[r] = i;
    }

    last[r] = i;
    if ((r == 0) && (last[r] > *right - *left)
        || (last[r] - first[r] > *right - *left)) {
      *right = last[r];
      *left = ((r == 0) ? 0 : first[r]);
    }
  }
}

int main() {
  FILE *fin, *fout;
  int n, c, i, j, l, r, steps;

  fin = fopen("struguri.in", "rt");
  fout = fopen("struguri.out", "wt");

  fscanf(fin, "%d %d", &c, &n);
  for (i = 1; i <= n; i++) {
    fscanf(fin, "%d", &v[i].x);

    v[i].idx = i;
  }

  if (c == 1) {
    l = r = 0;
    compute(v, n, &l, &r);

    // se afiseaza lungimea secventei de gramezi, numarul de ordine al primei
    // gramezi din secventa, respectiv numarul de ordine al ultimei gramezi
```

```
    // din secventa
    fprintf(fout, "%d %d %d\n", v[r].idx - v[l].idx, v[l + 1].idx, v[r].idx);
  } else {
      steps = 0;
      // cat timp mai sunt elemente in vectorul v
      while (n) {
        l = r = 0;
        // se determina cea mai lunga subsecventa a.i. suma valorilor
            elementelor
        // sa fie un multiplu de n
        compute(v, n, &l, &r);

        // se salveaza valorile pentru afisarea raspunsului
        // numarul de carute care transporta struguri in tura respectiva
        a[steps].n = n;
        // cantitatea de struguri pe care o transporta fiecare caruta
        a[steps].carry = (s[r] - s[l]) / n;
        // lungimea secventei de gramezi / numarul de gramezi transportate
        a[steps].m = r - l;
        a[steps].t = (int*)malloc(sizeof(int) * (r - l));

        for (i = l + 1; i <= r; i++) {
          // numarul de ordine ale gramezii transportate in tura curenta
          a[steps].t[i - (l + 1)] = v[i].idx;
        }
        steps++;

        // se sterg gramezile transportate in tura curenta
        for (i = l + 1, j = r + 1; j <= n; i++, j++) {
          v[i] = v[j];
        }
        // se actualizeaza numarul de gramezi ramase pentru transport
        n -= (r - l);
      }

      // se afiseaza raspunsul
      fprintf(fout, "%d\n", steps);
      for (i = 0; i < steps; i++) {
        fprintf(fout, "%d %d %d ", a[i].n, a[i].carry, a[i].m);
        for (j = 0; j < a[i].m; j++) {
          fprintf(fout, "%d ", a[i].t[j]);
        }
        fprintf(fout, "\n");
      }
  }

  fclose(fin);
  fclose(fout);

  return 0;
}
```

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introducere în Algoritmi*, Computer Libris Agora, Cluj-Napoca, 1999.

[2] M. Coşulschi, M. Gabroveanu, *Practica programării în C*, Editura Universitaria, Craiova, 2014.