Малко контролно II

Име: ФН: Курс: Група:

A) All Paths *(6 точки)*: Предложете алгоритъм, който намира броя на всички ациклични пътища между два върха в свързан граф.

B) SubArray : Даден е масив от n *ограничени* цели числа. Предложете алгоритъм с време $O(n)$, който намира най-голямата сума на подмасив с $k \leq n$ елемента, ако:

b1) елементите са произволни. *(3 точки)*

b2) елементите са последователни. *(3 точки)*

C) 3 Knights *(6 точки)*: Шахматна дъска е индексирана от 0 до 7 по хоризонтал и вертикал. На полета (x_1, y_1), (x_2, y_2) и (x_3, y_3) са поставени 3 шахматни коня. Един след друг те правят съответно k_1, k_2 и k_3 хода, след което всички се намират на едно и също поле.

Предложете алгоритъм, който намира поле, за което това става с възможно най-малко ходове. Също така алгоритъмът трябва да намира k_1, k_2 и k_3.

D) Draw : В турнир по футбол с четен брой отбори - n се провежда жребий за първи кръг.

d1) Предложете алгоритъм с време $O(n)$, който намира броя на различните жребии без оглед кой отбор е домакин и кой е гост. *(4 точки)*

d2) Предложете алгоритъм, който намира всички възможни жребии за турнир с $n \leq 12$ отбора, индексирани от 0 до $n-1$. *(6 точки)*

Забележка: Условието, че числата са ограничени в задачата SubArray беше казано по време на самото контролно. С цел яснота на цялото условие за бъдещи читатели, тук то е променено.

Също така възникнаха въпроси какво означават *произволни* и *последователни* елементи на масив.

k последователни елементи на масив $a[n]$, представлява редица от вида:

$$a[i], a[i+1], \ldots, a[i+k-1], \text{ където } 0 \leq i \leq n-k$$

k произволни елементи от масив $a[n]$, представялва редица от вида:

$$a[i_1], a[i_2], \ldots, a[i_k], \text{ където } 0 \leq i_1, \ldots, i_k \leq n-1 \text{ и } i_\alpha \neq i_\beta \text{ за } 1 \leq \alpha < \beta \leq k$$

Решения (псевдокодове):

A) All Paths

```
initialize visited[n] with false;

int all_paths(node s, node t)
{
      if (s == t) return 1;
      visited[s] = true;
      int sum = 0;
      for (node v in adj[s])
            if (visited[v] == false) sum += all_paths(v, t);
      visited[s] = false;
      return sum;
}
```

B) SubArray

b1) За произволни елементи

Очевидно търсеният подмасив съдържа първите k най-големи елемента на масива, така че сортираме масива с бърз алгоритъм за сортиране на цели числа (можем да приложим такъв, тъй като знаем, че числата са ограничени) и намираме сумата на последните k елемента.

b2) За последователни елементи

```
int a[n];

int subarr(int k)
{
      int s = 0, m;
      for (int i = 0; i < k; i++) s += a[i];
      m = s;

      for (int i = k; i < n; i++)
      {
            s += (a[i] - a[i - k]);
            if (s > m) m = s;
      }

      return m;
}
```

C) 3 Knights

Нека дъската представлява граф, с върхове полетата и ребра, двойки полета, за които може да се отиде от едното до другото за един ход на коня.

Правим BFS на цялата дъска за всеки кон и търсим поле (x, y), което минимизира $dist_1(x, y) + dist_2(x, y) + dist_3(x, y)$.

Съответно $k_1 = dist_1(x, y)$, $k_2 = dist_2(x, y)$, $k_3 = dist_3(x, y)$.

D) Draw

d1) Намиране на брой жребии

Нека T_n е броят различни жребии за n отбора. Фиксираме първия отбор и избираме негов опонент. Това, разбира се, може да стане по $n - 1$ начина. Аналогично разпределяме и останалите отбори. Така получаваме, че $T_n = (n - 1).T_{n-2}$ и $T_2 = 1$ (за два отбора има само един възможен жребий). Това определя общата формула за броя жребии: $(n - 1)!!$

```
int s = 1;
for (int i = 3; i < n; i += 2) s *= i;
print(s);
```

d2) Жребии в явен вид

Тук логиката е абсолютно същата като в d1). Ако са останали 2 неразпределени отбора, то те трябва да играят помежду си, така че извеждаме целия жребий.

В противен случай - избираме опонент на първия отбор и разпределяме останалите.

```
int a[12] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
int c = 1;
int n; // това е броят отбори

function gen_draw(int i)
{
	if (i == n - 2)
	{
		print(c++, ": ");
		for (int k = 0; k < n; k += 2)
			print("(", a[k], " - ", a[k + 1], ") ");
		printnl;
	}
	else for (int j = i + 1; j < n; j++)
	{
		swap(a[i + 1], a[j]);
		gen_draw(i + 2);
		swap(a[i + 1], a[j]);
	}
}

gen_draw(0);
```