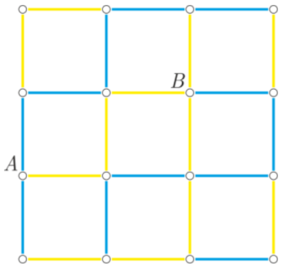
Малко контролно II

Име: курс: гр. ФН:

Задача 1. *(8 точки)*

Някои от отсечките в квадратна мрежа с размер $n \times n$ са сини, а другите - жълти (виж картинката).



Предложете алгоритъм с време $O(n^4)$, който намира дали в мрежата има път (може да е и цикличен) между два възела A и B, който съдържа повече сини отколкото жълти отсечки.

Задача 2. *(12 точки)*

Предложете алгоритъм с време $O(n^2)$, който намира броя на векторите от цели числа $\langle a_1, \dots, a_n \rangle$, такива че $|a_1| + \dots + |a_n| = n$.

Задача 3. *(8 точки)*

Предложете алгоритъм с време $O(n\sqrt{n})$, който намира броя на различните мултимножества M от положителни цели числа, такива че:

$$\sum_{x \in M} x^2 = n$$

Примерни решения:

Задача 1.

Да разгледаме тегловния граф породен от мрежата, с тегла -1 за синя отсечка и 1 за жълта. Този граф има n^2 върха и $\Theta(n^2)$ ребра. Изпълняваме алгоритъма на Белман-Форд за връх A. Това става за време $O(n^4)$.

Ако алгоритъмът намери отрицателен цикъл, то отговорът на въпроса е ДА - въртим се около отрицателния цикъл достатъчно пъти и после отиваме към връх B.

В противен случай, отговорът е ДА, само ако $dist[B]$ е отрицателно число.

Задача 2.

Нека $S(n,k)$ е броят на n-мерните вектори, със сума от модули на компонентите k.

Лесно се вижда, че:

$$S(n,k) = S(n-1,k) + 2\sum_{i=1}^{k} S(n-1,k-i)$$

Фиксираме първата компонента. Ако е 0, то търсим $S(n-1,k)$, тъй като 0 не променя сумата от компонентите. В другите случаи имаме 2 варианта - съответно за положително и за отрицателно число.

От този израз изваждаме съответния за $S(n,k-1)$ и получаваме:

$$S(n,k) = S(n-1,k) + S(n,k-1) + S(n-1,k-1)$$

Граничните случаи са: $S(i,0) = 1$, за всяко i и $S(0,j) = 0$ за всяко $j > 0$.

Динамична таблица за $S(n,k)$ се попълва за време $O(nk)$ или в случая за $O(n^2)$.

Псевдокод:

```
function NUM_OF_VECTORS(n)
{
      S[][] = new Array of size (n+1)×(n+1)

      for (i = 0; i <= n; i++) S[i][0] = 1;
      for (j = 1; j <= n; j++) S[0][j] = 0;

      for (i = 1; i <= n; i++) for (j = 1; j <= n; j++)
      {
            S[i][j] = S[i][j-1] + S[i-1][j] + S[i-1][j-1];
      }

      return S[n][n];
}
```

Задача 3.

Разглеждаме функцията:

$D(i,j) =$ „броят мултимножества за i, в които всеки елемент е по-голям или равен на j".

В зависимост от това дали j може да участва в мултимножествата, тази функция се изчислява така:

$$D(i,j) = \begin{cases} D(i-j^2,j) + D(i,j+1) & ,j^2 \leq i \\ D(i,j+1) & ,j^2 > i \end{cases}$$

Граничните случаи са: $D(0,j) = 1$ за всяко j и $D(i,j) = 0$ при $j = 1 + \lfloor\sqrt{n}\rfloor$ и $1 \leq i \leq n$.

Търсеният резултат е $D(n,1)$. С помощта на динамична таблица с $\Theta(\sqrt{n})$ колони, той се изчислява по горната формула за време $O(n\sqrt{n})$.

Псевдокод:

```
function SUM_OF_SQUARES(n)
{
      s = Floor(Sqrt(n))
      D[][] = new Array of size (n+1)×(s+2)

      for (j = 0; j <= s + 1; j++) D[0][j] = 1;
      for (i = 1; i <= n; i++) D[i][s+1] = 0;

      for (j = s; j >= 1; j--) for (i = 1; i <= n; i++)
      {
            D[i][j] = D[i][j+1];
            if (j*j <= i) D[i][j] += D[i-j*j][j];
      }

      return D[n][1];
}
```