Algorytmy i Struktury Danych, 9. ćwiczenia

2021-12-01

Contents

1 Klasówka 2021 (1), zadanie 1

Powiemy, że zbiór n liczb całkowitych jest prawie gęsty, jeśli zawiera podzbiór o rozmiarze większym niż $n/3$, w którym różnica pomiędzy największym i najmniejszym elementem jest mniejsza od n. Taki podzbiór nazywamy świadectwem. Dana jest dodatnia liczba całkowita n oraz n-elementowy zbiór liczb całkowitych S. Zaproponuj algorytm, który w czasie liniowym sprawdzi, czy S jest prawie gęsty.

Uwaga: 3 punkty uzyskasz za algorytm, który w czasie liniowym sprawdza, czy wskazany, dowolny element z S należy do jakiegoś świadectwa.

Rozwiązanie:

Algorithm 1: CzyNależy(S, n, x)

niech $S' = \{e \in S : |e - x| < n\}$
ponieważ wszystkie elementy z S' należą do przedziału $[e - n, \ldots, e + n]$ długości $2n + 1$ stąd możemy posortować S' w czasie liniowym
$S'' = sorted(S')$
for $i = 1, \ldots, m - \lceil \frac{n}{3} \rceil + 1$ **do**
    $j := i + \lceil \frac{n}{3} \rceil - 1$
    if $S''[j] - S''[i] < n$ **then**
        return *TAK* (ponieważ $s_i, \ldots, s_j$ + być ewentualnie x są świadectwem)
return *NIE*

Algorithm 2: Rozwiązanie(S, n)

foreach $k \in \{\lceil n/4 \rceil, \lceil n/2 \rceil, \lceil 3n/4 \rceil\}$ **do**
    $x :=$ DeterministicSelectKthElement(S, k)
    if *CzyNależy*(S, n, x) **then**
        return *TAK*
return *NIE*

2 Klasówka 2021 (1), zadanie 2

Zaproponuj optymalny ze względu na porównania algorytm sortowania siedmioelementowego ciągu $x_1, \ldots, x_7$, o którym wiadomo, że $x_1 < x_2$, $x_1 < x_3$, $x_4 < x_5$, $x_4 < x_6$. Udowodnij optymalność swojego algorytmu.

Rozwiązanie: Wszystkich permutacji 7 elementowych spełniających warunki zadania jest:

$$\binom{7}{3} \cdot 2 \cdot \binom{4}{3} \cdot 2 = 35 \cdot 2 \cdot 4 \cdot 2 = 560$$

Ponieważ:

- $\{x_1, x_2, x_3\}$ możemy wybrać z 7 elementów na $\binom{7}{3}$ sposobów
- mamy dwie możliwości na porównanie x_2 i x_3 ($x_2 < x_3$ lub $x_2 > x_3$)
- $\{x_4, x_5, x_6\}$ możemy wybrać z pozostałych 4 elementów na $\binom{4}{3}$ sposobów,
- mamy dwie możliwości na porównanie x_5 i x_6 ($x_5 < x_6$ lub $x_5 > x_6$)

Algorithm 3: Rozwiązanie

cmp(x_2, x_3) (bez straty ogólności $x_2 < x_3$)
cmp(x_5, x_6) (bez straty ogólności $x_5 < x_6$)
wstaw x_7 pomiędzy $x_1 < x_2 < x_3$ używając 2 porównań
scal dwa uporządkowane ciągu (4 i 3 elementowe) przy użyciu 6 porównań

3 Klasówka 2021 (1), zadanie 3

Dla dodatniej liczby całkowitej n kratownicą M_n nazywamy skierowany graf (V, E) bez pętli, w którym $V = \{(x, y) : x = 0, 1, ..., n \text{ oraz } y = 0, 1, ..., n\}$ i

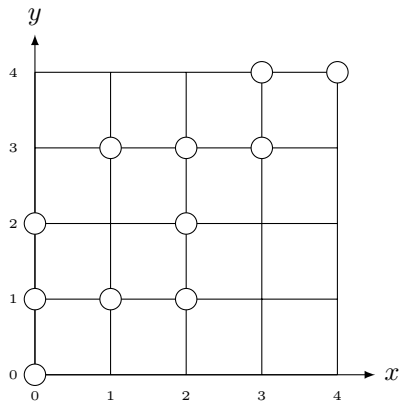
$$E = \{(x, y) \to (x', y') : 0 \leq x' - x \leq 1 \text{ oraz } 0 \leq y' - y \leq 1\}.$$

Wierzchołki grafu M_n pomalowano na biało lub czarno. Białą ścieżką nazwiemy każdą ścieżkę, na której wszystkie wierzchołki są białe. Dane są liczba całkowita $n > 0$, nieujemna liczba całkowita $m \leq (n+1)^2$ oraz m różnych, białych wierzchołków $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$. Pozostałe wierzchołki są czarne. Zaproponuj wydajny (czasowo i pamięciowo) algorytm, który obliczy liczbę wszystkich białych ścieżek z wierzchołka $(0, 0)$ do wierzchołka (n, n).

Rozwiązanie: Aby uprościć rozumowanie załóżmy, że wszystkie operacje arytmetyczne możemy wykonywać w czasie stałym. Jest to o tyle istotne, że wynikowa wartość może być większa niż $\binom{2n}{n}$ (to jest liczba ścieżek dla pełnego białego grafu M_n w których poruszamy się tylko w prawo lub w górę).
Przykład:
Dla $n = 4$ i $m = 11$ białych wierzchołków:

$$B = \{(0,0), (0,1), (0,2), (1,1), (1,3), (2,1), (2,2), (2,3), (3,3), (3,4), (4,4)\}$$

mamy następujący graf:



Niech $Ile(i,j)$ oznacza liczbę białych ścieżek z $(0,0)$ do (i,j) spełniających warunki zadania.

Jeśli $m = \Theta(n^2)$ to możemy w czasie $O(n^2)$ policzyć $Ile(n,n)$ korzystając z następujacych wzorów:

$$Ile(0,0) = \begin{cases} 1 & \text{jeśli wierzchołek } (0,0) \text{ jest biały} \\ 0 & wpp \end{cases}$$

$$Ile(\cdot,-1) = Ile(-1,\cdot) = 0$$

Jeśli (i,j) jest biały to:

$$Ile(i,j) = Ile(i-1,j) + Ile(i-1,j-1) + Ile(i,j-1)$$

Jeśli (i,j) jest czarny to:

$$Ile(i,j) = 0$$

Jeśli $m < n$ to odpowiedzią jest 0 (nie ma wystarczająco dużo białych węzłów).

Jeśli $m = \Omega(n)$ to możemy policzyć $Ile(n,n)$ w czasie $O(m)$. Załóżmy, że wierzchołki $(0,0)$ i (n,n) są białe (jeśli byłoby inaczej to oczywiście odpowiedzią jest 0).
Zdefiniujmy sobie 3 porządki ($<_{x,y}$, $<_{y,x}$, $<_d$) na parach liczb:

- $(x_1,y_1) <_{x,y} (x_2,y_2)$ wtw $(x_1 < x_2)$ lub $(x_1 = x_2) \wedge (y_1 < y_2)$
- $(x_1,y_1) <_{y,x} (x_2,y_2)$ wtw $(y_1 < y_2)$ lub $(y_1 = y_2) \wedge (x_1 < x_2)$
- $(x_1,y_1) <_d (x_2,y_2)$ wtw $(x_1 - y_1 < x_2 - y_2)$ lub $(x_1 - y_1 = x_2 - y_2) \wedge (x_1 < x_2)$

W pierszym kroku uporządkujmy leksykograficznie (czyli wg. $<_{x,y}$) ciąg B. Od teraz zakładamy, że:

$$(x_1,y_1) <_{x,y} (x_2,y_2) <_{x,y} \ldots <_{x,y} (x_m,y_m)$$

Przy takim uporządkowaniu definiujemy ILE$[i]$ jako liczbę białych ścieżek z $(0,0)$ do (x_i,y_i).

Dodatkowo definiujemy pomocnicze wartości:

$$\textsc{GdzieX}[i] = \begin{cases} j & \text{jeśli } (x_j = x_i) \wedge (y_j + 1 = y_i) \\ -1 & \text{jeśli takie } j \text{ nie istnieje} \end{cases}$$

$$\textsc{GdzieY}[i] = \begin{cases} j & \text{jeśli } (y_j = y_i) \wedge (x_j + 1 = x_i) \\ -1 & \text{jeśli takie } j \text{ nie istnieje} \end{cases}$$

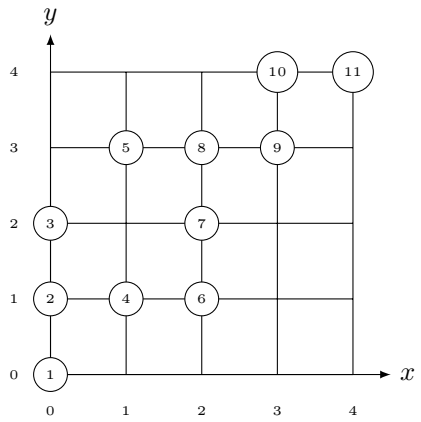
$$\textsc{GdzieD}[i] = \begin{cases} j & \text{jeśli } (x_j + 1 = x_i) \wedge (y_j + 1 = x_i) \\ -1 & \text{jeśli takie } j \text{ nie istnieje} \end{cases}$$

Wartości $\textsc{Gdzie}(X/Y/D)$ można obliczyć w czasie $O(m)$ sortując wierzchołki wg porządków $<_{x,y}$ / $<_{y,x}$ / $<_d$ (co możemy zrobić w czasie $O(m)$). Jedynym kandydatem na wartość Gdzie jest poprzedni wierzchołek w danym porządku (co możemy zweryfikować w czasie $O(1)$).
Gdy już mamy obliczone wartości Gdzie, wartości *Ile* obliczamy korzystając ze wzoru:

$$\textsc{Ile}[i] = \textsc{Ile}[GdzieX[i]] + \textsc{Ile}[GdzieY[i]] + \textsc{Ile}[GdzieD[i]]$$

(przy czym sztucznie definiujemy $\textsc{Ile}[-1] = -0$)

Przykład:



i	1	2	3	4	5	6	7	8	9	10	11
(x_i, y_i)	(0,0)	(0,1)	(0,2)	(1,1)	(1,3)	(2,1)	(2,2)	(2,3)	(3,3)	(3,4)	(4,4)
$\textsc{GdzieX}[i]$	-1	1	2	-1	-1	-1	6	7	-1	9	-1
$\textsc{GdzieY}[i]$	-1	-1	-1	2	-1	4	-1	5	8	-1	10
$\textsc{GdzieD}[i]$	-1	-1	-1	1	3	-1	4	-1	7	8	9
$\textsc{Ile}[i]$	1	1	1	2	1	1	3	4	7	11	18