ASD - ćwiczenia V

Zadanie o „ustawionym“ teleturnieju

Pan Gerwazy ma zamiar wystąpić w teleturnieju „Jaki to algorytm?“. Niestety jego wiedza z dziedziny algorytmów i struktur danych jest bardzo uboga, co w warunku uczciwej rozgrywki, wyklucza go z grona zwycięzców. Jednak los Pana Gerwazego nie jest jeszcze przesądzony – tak się składa, że jego stryjeczny brat, o wdzięcznym imieniu Protazy, układa wszystkie pytania do rozważanej zabawy. Panowie doszli więc do wniosku, że przy odrobinie „szczęścia“ (a wiadomo, że szczęściu trzeba pomagać), zwycięzcą następnego odcinka teleturnieju „Jaki to algorytm?“ zostanie pan Gerwazy.

Teleturniej „Jaki to algorytm“ składa się z k rund, z których w każdej zawodnikowi zadawane są trzy pytania z puli n pytań przeznaczonych na daną rundę. Zatem, łącznie do teleturnieju przygotowywane jest dokładnie $k \cdot n$ pytań. Pan Protazy dostarczył panu Gerwazemu odpowiednio dwa segregatory $Q = \{Q[1], Q[2], \ldots, Q[k]\}$ oraz $A = \{A[1], A[2], \ldots, A[k]\}$ po k zeszytów każdy, z których zeszyt $Q[i]$ zawiera pytania do i-tej rundy teleturnieju a zeszyt $A[i]$ zawiera odpowiedzi do pytań z i-tej rundy teleturnieju. Tuż przed przekazaniem zeszytów panu Gerwazemu, Protazy nie oparł się wyrzutom sumienia i postanowił odrobinę utrudnić zadanie swojemu stryjecznemu bratu. Z każdego zeszytu $Q[i]$ jak i z każdego zeszytu $A[i]$ usunął pewną liczbę p pytań bądź q odpowiedzi, gdzie $0 \leq p \leq n$ oraz $0 \leq q \leq n$. Dodatkowo dla utrudnienia nauki pytań i odpowiedzi, losowo poprzekładał zawartość każdego z zeszytów, tj. pytania jak i odpowiedzi w zeszytach są ułożone w dowolnej kolejności. Gdy tak zmodyfikowane „pomoce“ trafiły do rąk pana Gerwazego, ten zaczął zastanawiać się, czy warto brać udział w tym teleturnieju. A ponieważ jest człowiekiem bardzo pragmatycznym, ustalił, że jeżeli dostarczone segregatory pytań i odpowiedzi pozwolą mu na wygranie, czyli udzielenie poprawnej odpowiedzi na wszystkie $3 \cdot k$ pytania w co najmniej połowie z możliwych wariantów rozgrywki teleturnieju „Jaki to Algorytm“, to warto spróbować. Niestety tak jak wspomnieliśmy, Pan Gerwazy jest raczej laikiem w dziedzinie algorytmów, nie wie jak szybko sprawdzić przedstawiony warunek dla rozważanych segregatorów Q i A.

Opracuj funkcję

```
bool START(FileBinder Q[], FileBinder A[], int k, int n),
```

możliwie asymptotycznie najszybszą, która zgodnie z warunkiem wymyślonym przez pana Gerwazego, wskaże mu właściwą decyzję ($TRUE$ – start w teleturnieju, $FALSE$ – w p.p.). **Uwaga:** przyjmujemy, że kolejność zadawania pytań w trakcie każdej rundy jest istotna i tworzy rozróżnialne warianty rozgrywki.

- Podaj słowny opis procedury `START()`.
- Napisz pseudokod procedury `FIND()` (w kodzie procedury można używać dowolnych algorytmów przedstawionych na wykładzie, bez potrzeby wypisywania ich implementacji, należy jednak wyjaśnić jak i do czego się ich używa).
- Oszacuj złożoność czasową algorytmu, przy czym dokładnie opisz składniki funkcji złożoności, jeżeli procedura składa się z kilku faz. Nie zapomnij precyzyjnie określić co stanowi argument funkcji złożoności.

Zakładamy, że segregatory i zeszyty są reprezentowane przez poniższe typy danych:

```
typedef NoteBook [] FileBinder;
typedef str_NoteBook NoteBook;
typedef str_Element Element;

struct str_NoteBook {
```

```
  int s;                              // liczba pytań/odpowiedzi
  Element E[];                        // tablica pytań/odpowiedzi
};

struct str_Element {
  int index;                          // indeks pytania/odpowiedzi
  char word[];                        // treść pytania/odpowiedzi
};
```

Podsumowanie wiedzy z zakresu problemu sortowania

Zadania

1. Co oznacza termin "działanie w miejscu" w odniesieniu do algorytmów sortowania przez porównania? Które ze znanych Ci algorytmów działają w miejscu, a które nie działają w miejscu?

2. Co oznacza termin "stabilność" w odniesieniu do algorytmów sortowania przez porównania? Podaj przykład danych, dla których własność stabilności algorytmu sortującego jest istotna. Który ze znanych Ci algorytmów sortowania (zgodnie z schematami przedstawionymi na wykładzie) jest stabilny?

3. Niech $S_1, S_2, \ldots, S_k$ będą zbiorami składającymi się z liczb całkowitych zapisanych w systemie dziesiętnym oraz spełniają warunek $\forall i \in \{1, 2, \ldots, k\} : |S_i| = n$. Zaproponuj metodę postępowania dla problemu sortowania wszystkich możliwych sum postaci $r_1 + r_2 + \ldots + r_k$, gdzie $r_i \in S_i$ jeżeli uznajemy, że operacją dominującą dla proponowanego rozwiązania jest porównywanie liczb całkowitych oraz:

   (a) określono warunek: $W(n,k) = O\left(n^{2k}\right)$ i $A(n,k) = O\left(kn^k \log(n)\right)$,

   (b) określono warunek: $W(n,k) = O\left(kn^k \log(n)\right)$ i $A(n,k) = O\left(kn^k \log(n)\right)$,

   (c) określono warunek: $W(n,k) = O\left(kn^k \log(n)\right)$, $A(n,k) = O\left(kn^k \log(n)\right)$ i metoda ta ma działać w miejscu,

   (d) załóżmy dodatkowo, że $\forall i \in \{1, 2, \ldots, k\} : \forall j \in \{1, 2, \ldots, n\} : \lceil log_{10}(|S_i[j]|) \rceil \leq d$ (inaczej mówiąc dowolny element dowolnego zbioru S_i jest dodatnią albo ujemną liczbą co najwyżej d-cyfrową). Określono warunek: $W(n,k,d) = O\left((d+k)\, n^k\right)$, $A(n,k,d) = O\left((d+k)\, n^k\right)$,

   (e) jaka byłaby złożoność pesymistyczna i oczekiwana twoich rozwiązań jeżeli, zamiast wszystkich sum postaci $r_1 + r_2 + \ldots + r_k$ rozważymy wszystkie iloczyny postaci $r_1 \cdot r_2 \cdot \ldots \cdot r_k$, gdzie $r_i \in S_i$.

4. Przedstaw kolejne etapy (stan pomocniczej tablicy w kolejnych iteracjach pętli głównej) sortowania danego ciągu za pomocą algorytmu CountingSort:

   (a) 4, 6, 3, 2, 5, 6, 2, 3, 2,

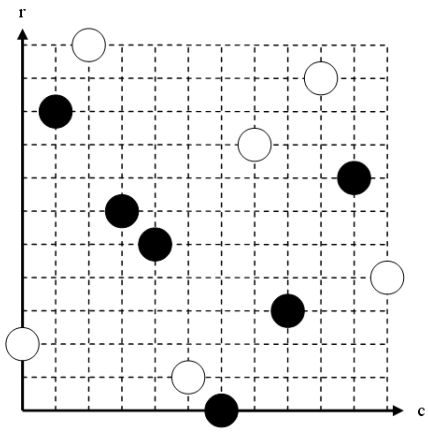
   (b) 6, 5, 8, 5, 2, 2, 5, 6.

Zadanie o dziecinnej grze w „liniowe kropki" (do domu)

Do gry w liniowe kropki potrzebna jest plansza składająca się z $n \times n$ pól (n wierszy na n kolumny), zbiór $\frac{n}{2}$ kropek białych $W = W[1], W[2], \ldots, W\left[\frac{n}{2}\right]$ oraz zbiór $\frac{n}{2}$ kropek czarnych $B = B[1], B[2], \ldots, B\left[\frac{n}{2}\right]$. Każde pole, może być w danej chwili zajmowane przez co najwyżej jedną kropkę białą albo czarną. Na początku rozgrywki w losowy sposób rozmieszcza się kropki z obu zbiorów na planszy do gry, w taki sposób, że żadne dwie kropki ze zbioru $W \cup B$ nie leżą na dwóch polach planszy o tej samej współrzędnej wierszowej albo kolumnowej. Następnie gracze postępują zgodnie z poniższymi zasadami:

- grę rozpoczyna gracz z białymi kropkami,
- gracze grają na przemian,
- w każdym ruchu dowolny z graczy może przenieść dokładnie jedną ze swoich kropek leżącą na polu o współrzędnych (r, c) jedno z pól (oczywiście mieszczących się na planszy):
  - $(r+1, c)$ – ruch w górę,
  - $(r+1, c+1)$ – ruch w górę i w prawo,
  - $(r, c+1)$ – ruch w prawo,
  - $(r-1, c+1)$ – ruch w dół i w prawo,
  - $(r-1, c)$ – ruch w dół,
  - $(r-1, c-1)$ – ruch w dół i w lewo,
  - $(r, c-1)$ – ruch w lewo,
  - $(r+1, c-1)$ – ruch w górę i w lewo,

  pod warunkiem, że pole to nie jest okupowane przez inną kropkę ze zbioru $W \cup B$,
- przeniesienie kropki z pola o współrzędnych (r, c) na pole o współrzędnych (r', c') uprawnia właściciela tej kropki do usunięcia z planszy wszystkich kropek przeciwnika, których współrzędne wierszowe są równe r' albo kolumnowe są równe c'.
- wygrywa ten, kto pierwszy usunie wszystkie kropki swojego rywala.



Rysunek 1: Przykładowy początkowy stan planszy dla $\frac{n}{2} = 6$.

Zaprojektuj możliwie efektywną funkcję

`int BEST_MOVE(Dot W[], Dot B[], int n)`,

która dla danego początkowego ustawienia kropek białych W i czarnych B na planszy gry, wyznaczy maksymalną liczbę kropek czarnych, jaką gracz posiadający kropki białe, może usuną z planszy w pierwszym ruchu. Zakładamy, że kropki są reprezentowane za pomocą zmiennej typu `Dot` postaci:

```
typedef str_Dot Dot;

struct str_Dot {
  int r, c;
};
```

gdzie atrybuty r, c określają odpowiednio aktualną współrzędną wierszową i kolumnową kropki na planszy gry.