

# Algorytmy online

## Lista EXTRA

### 1 Zadania niewykorzystane

**Zadanie** Pokaż, że FIFO jest nie jest algorytmem zaznaczającym. Pokaż, że FIFO jest  $k$ -konkurencyjny.

**Zadanie** Problem szukania krowy można też zdefiniować w następujący sposób. Mamy robota umieszczonego początkowo w punkcie  $(0, 0)$  i prostą o równaniu  $x = a$ , gdzie  $a$  jest nieznane robotowi. Należy jak najszybciej znaleźć tę prostą (tj. przejść przez nią w dowolny sposób). Oczywiście każdy rozsądny robot będzie szukać jej tylko wzdłuż osi OX.

Rozważmy teraz modyfikację tego problemu, w której wiemy, że prosta ma równanie  $x = a$  albo  $y = a$ , tj. wiemy, że jest równoległa do jednej z osi układu współrzędnych, ale nie wiemy do której. Liczba  $a$  jest nieznana robotowi, ale można założyć, że jest liczbą całkowitą. Oczywiście  $\text{OPT} = a$ .

Skonstruuj deterministyczny algorytm, który chodzi tylko po osiach OX lub OY, którego współczynnik konkurencyjności wynosi co najwyżej 25.

**Zadanie** Dla problemu z poprzedniego zadania, skonstruuj deterministyczny algorytm (niekoniecznie chodzący po osiach OX i OY) o współczynniku konkurencyjności co najwyżej 14.

**Zadanie** Pokaż, dolne ograniczenie  $2 + \frac{1}{2D}$  na konkurencyjność dowolnego randomizowanego algorytmu ALG. Wskazówka: załóż że ALG jest zdefiniowany w każdym kroku jako rozkład prawdopodobieństwa. Na generowanym ciągu wejściowym uruchom jednocześnie ALG, EDGE i OPT starając się krzywdzić ALG nie mniej niż EDGE i jednocześnie gwarantować, że koszt EDGE jest odpowiednio wysoki w stosunku do OPT.

**Zadanie** Rozważmy następujący algorytm CNT dla problemu przenoszenia pliku w klice. Z każdym wierzchołkiem  $v$  CNT wiąże licznik  $c_v$ , początkowo równy 0. Algorytm ma dwa tryby działania: *tryb zwykły* i *tryb przeczekiwania*.

- W trybie zwykłym jeśli żądanie jest w wierzchołku  $u$ ,  $c_u$  jest zwiększany o 1. Następnie jeśli  $c_u = D$ , plik jest przenoszony do  $u$ , licznik  $c_u$  jest ustawiany na 0 i algorytm przechodzi do trybu przeczekiwania.
- W trybie przeczekiwania algorytm oczekuje  $D$  odwołań do wierzchołków innych niż ten, który ma plik. Żadne liczniki nie są zmieniane podczas tego trybu.

Pokaż, że algorytm CNT jest 3-konkurencyjny. Wskazówka: podziel całą sekwencję na fazy; jedna faza to  $D$  odwołań do określonego wierzchołka + kroki w trybie przeczekiwania po przeniesieniu pliku do tego wierzchołka. Zauważ, że każdy krok należy do dokładnie jednej fazy.

**Zadanie** Pokaż, że konkurencyjność dowolnego deterministycznego algorytmu dla problemu szeregowania zadań na  $m$  niepowiązanych maszynach wynosi co najmniej  $\Omega(\log m)$ .

**Zadanie** Pokaż, że dla każdego  $\gamma$ -konkurencyjnego algorytmu dla problemu szeregowania zadań i dla dowolnie małego  $\epsilon > 0$ , istnieje algorytm który jest ściśle  $(\gamma + \epsilon)$ -konkurencyjny.

**Zadanie** Pokaż, że na podstawie algorytmu  $EXP_\lambda$  dla problemu minimalizacji obciążenia (z wykładu), można skonstruować algorytm, który będzie  $O(\log m)$  konkurencyjny. Wskazówka: Kiedy koszt OPT przekracza  $\lambda$ , podwajaj  $\lambda$ .

**Zadanie** Pokaż, że w następującym grafie skierowanym  $G$ , dolne ograniczenie na (ściśle) konkurencyjność deterministycznego algorytmu dla problemu minimalizacji obciążenia wynosi  $\Omega(\log m)$ .  $G$  składa się z  $\binom{k}{2}$  wierzchołków źródłowych  $s_{i,j}$ ,  $1 \leq i \neq j \leq k$ ,  $k$  wierzchołków pośrednich  $u_i$ ,  $1 \leq i \leq k$  i jednego ujścia  $t$ . Dowolny wierzchołek źródłowy  $s_{i,j}$  jest połączony z  $u_i$  i  $u_j$ , a dowolny wierzchołek pośredni  $u_i$  jest połączony z ujściem  $t$ . W takim grafie  $m = 2 \cdot \binom{k}{2} + k$ . Wskazówka: idea rozwiązania jest dokładnie taka sama jak w przypadku dolnego ograniczenia na problem szeregowania ograniczonych zadań.

**Zadanie** Rozważmy następujący uogólniony problem szeregowania zadań na  $m$  procesorach: z zadaniem  $t$  jest związany wektor  $m$  liczb  $(b_t(1), b_t(2), b_t(3), \dots, b_t(m))$ , gdzie  $b_t(i)$  jest kosztem przyporządkowania do procesora  $i$  (czasem wykonania na procesorze  $i$ ). Rozważmy następujący algorytm  $A_\lambda$ , który do wykonania zadania  $t$  wybiera procesor  $i$  minimalizujący

$$a \frac{L_{t-1}(i) + b_t(i)}{\lambda} - a \frac{L_{t-1}(i)}{\lambda},$$

gdzie  $a = 3/2$  a  $L_{t-1}(i)$  oznacza obciążenie procesora  $i$  na początku kroku  $t$ .

Pokaż, że na sekwencji, której rozwiązanie optymalne ma koszt co najwyżej  $\lambda$ , koszt  $A_\lambda$  wynosi  $O(\log m) \cdot \lambda$ . Jak na tej podstawie skonstruować algorytm  $O(\log m)$ -konkurencyjny?

**Zadanie** Rozważ modyfikację algorytmu dla ułamkowego problemu pokrywania zbiorami podanego na wykładzie. Mianowicie podczas rozważania elementu  $e_j$  w kroku  $j$  przypisanie  $x_S \leftarrow (1 + 1/c_S) \cdot x_S + 1/(m \cdot c_S)$  zastępujemy przypisaniem

$$x_S \leftarrow \left(1 + \frac{1}{c_S}\right) \cdot x_S + \frac{1}{|\{S : e_j \in S\}| \cdot c_S}$$

zaś  $y_j$  zwiększany jest po prostu o 1. Pokaż, że (ściśle) konkurencyjność algorytmu zatrzymanego po  $k$  krokach można ograniczyć przez  $O(\log d)$ , gdzie  $d = \max_{1 \leq j \leq k} |\{S : e_j \in S\}|$ . Uwaga: generowane przez algorytm rozwiązania programu dualnego będą dopuszczalnymi rozwiązaniami, dopiero gdy podzielimy je przez  $\Omega(\log d)$ .

**Zadanie** Rozważmy następujący proces losowego zaokrąglania rozwiązania ułamkowego  $\{x_S\}_{S \in \mathcal{F}}$  dla problemu pokrywania zbiorami. Dla każdego zbioru  $S$  w kroku  $k$  losujemy liczby  $T_{S,j}$  z jednostajnym rozkładem z odcinka  $[0, 1]$ , gdzie  $j \in \{1, 2, \dots, \ell \cdot \lceil \ln(k+1) \rceil\}$  a  $\ell$  jest pewną stałą.<sup>1</sup> Algorytm randomizowany bierze zbiór  $S$  do rozwiązania jeśli  $x_S \geq \min_j \{T_{S,j}\}$ .

**Zadanie** Rozważamy problem Adwords w wariacie gdzie budżet każdego gracza (reklamodawcy) wynosi  $b = 1$ , gracz jest zainteresowany pewnymi słowami kluczowymi i jest za nie skłonny zapłacić 1, a za pozostałe nie chce płacić w ogóle. Innymi słowy: każdego gracza stać na jednokrotne wyświetlenie jego reklamy.

Pokaż, że dla dowolnego  $n$  i dowolnego algorytmu deterministycznego DET można stworzyć taką instancję wejściową, że OPT zyskuje na niej  $n$  zaś DET zyskuje co najwyżej  $\lceil n/2 \rceil$ .

**Zadanie** Rozważamy problem Adwords w wariacie gdzie budżet każdego gracza (reklamodawcy) wynosi  $b = 1$ , gracz jest zainteresowany pewnymi słowami kluczowymi i jest za nie skłonny zapłacić 1, a za pozostałe nie chce płacić w ogóle.

Rozważmy algorytm RANDOM, który dla danego słowa kluczowego wybiera losowego gracza, który ma jeszcze pieniądze i jest zainteresowany takim słowem kluczowym. Skonstruuj instancję, taką że OPT zyskuje na niej  $n$ , zaś oczekiwany zysk RANDOM wynosi co najwyżej  $n/2 + o(n)$ .<sup>2</sup>

**Zadanie** Pokaż dolne ograniczenie w wysokości  $\Omega(\log N)$  na ściśle konkurencyjność randomizowanego algorytmu routingu na linii zawierającej  $N + 1$  wierzchołków.

Wskazówka: skorzystaj z zasady minimaksowej i rozważ ciąg  $[0, N], [0, N/2], [N/2, N], [0, N/4], [N/4, N/2], [N/2, 3/4N], [3/4N, N], \dots, [0, 1], [1, 2], [2, 3], [3, 4], \dots, [N-2, N-1], [N-1, N]$ ; z pewnymi prawdopodobieństwami na wejściu pojawia się tylko prefiks tego ciągu.

<sup>1</sup>Łatwo osiągnąć monotoniczność generowanego rozwiązania (tj. własność, że algorytm nie usuwa z rozwiązania już przyjętych zbiorów): w każdym kroku doładowujemy brakujące liczby, ale nie zmieniamy już istniejących.

<sup>2</sup>Istnieje instancja dla których zysk RANDOM to  $n/2 + O(\log n)$ .

**Zadanie** Pokaż, że współczynnik (ściśle) konkurencyjności dowolnego algorytmu dla problemu szeregowania ograniczonych zadań na  $m$  procesorach wynosi co najmniej  $\Omega(\log m)$ .

**Zadanie** Weźmy dowolny graf w którym minimalna odległość między dwoma różnymi punktami wynosi  $m$  a maksymalna  $M$ . Pokaż, że jeśli istnieje  $k$ -konkurencyjny algorytm dla problemu pamięci podręcznej, to istnieje  $(k \cdot M/m)$ -konkurencyjny algorytm dla problemu  $k$  serwisantów w tym grafie.

*Marcin Bieńkowski*