Jakub Radoszewski — Treść zadania, Opracowanie
Juliusz Sompolski — Program

Dostępna pamięć: 32 MB. *OI, Etap II, dzień drugi, 12.02.2009*

Łyżwy

Bajtazar prowadzi klub łyżwiarski. Członkowie klubu spotykają się regularnie i razem trenują, przy czym korzystają zawsze z łyżew klubowych. Rozmiary łyżew są umownie numerowane od 1 do n. Każdy członek klubu ma pewien rozmiar stopy. U łyżwiarzy występuje jednakże współczynnik tolerancji d na rozmiar łyżew: łyżwiarz o rozmiarze stopy r może nosić łyżwy rozmiarów od r do r + d. Należy przy tym zaznaczyć, że ***żaden łyżwiarz nie zakłada nigdy jednocześnie dwóch łyżew różnych rozmiarów.***

Bajtazar zakupił na potrzeby klubu po k **par** *łyżew każdego z rozmiarów od 1 do n. W miarę upływu czasu nowe osoby zapisują się do klubu, a niektóre osoby wypisują się. Bajtazar martwi się, czy na każdych zajęciach będzie miał dla wszystkich członków klubu łyżwy odpowiedniego rozmiaru.*

Zakładamy, że początkowo nikt nie należy do klubu. Bajtazar dostarczył Ci sekwencję m zdarzeń postaci: przybyło/ubyło x członków klubu o rozmiarze stopy r. Bajtazar chciałby wiedzieć, po każdym takim zdarzeniu, czy ma łyżwy odpowiedniego rozmiaru dla wszystkich członków klubu. Poprosił Cię o napisanie programu, który to sprawdzi.

Wejście

Pierwszy wiersz standardowego wejścia zawiera cztery liczby całkowite n, m, k oraz d ($1 \leqslant n \leqslant 200\,000$, $1 \leqslant m \leqslant 500\,000$, $1 \leqslant k \leqslant 10^9$, $0 \leqslant d < n$)*, pooddzielane pojedynczymi odstępami i oznaczające odpowiednio: największy rozmiar łyżew, liczbę zdarzeń, liczbę par łyżew każdego rozmiaru zakupionych przez Bajtazara oraz tolerancję rozmiarową stóp łyżwiarzy. Kolejne m wierszy zawiera sekwencję m zdarzeń, po jednym w wierszu. Wiersz (i + 1)-szy (dla* $1 \leqslant i \leqslant m$*) zawiera dwie liczby całkowite:* r_i *oraz* x_i *(*$1 \leqslant r_i \leqslant n - d$, $-10^9 \leqslant x_i \leqslant 10^9$*), oddzielone pojedynczym odstępem. Jeśli* $x_i \geqslant 0$*, to oznacza to, że do klubu zapisało się* x_i *nowych członków o rozmiarze stopy* r_i*. Jeśli natomiast* $x_i < 0$*, to oznacza to, że z klubu wypisało się* $-x_i$ *członków o rozmiarze stopy* r_i*. Możesz założyć, że podana sekwencja zdarzeń ma sens, tzn. z klubu nie mogą wypisać się osoby, które się do niego nie zapisały.*

Wyjście

Twój program powinien wypisać na standardowe wyjście m wierszy. Wiersz i-ty (dla $1 \leqslant i \leqslant m$*) powinien zawierać jedno słowo* `TAK` *lub* `NIE`*, w zależności od tego, czy po i-tym zdarzeniu Bajtazar ma łyżwy odpowiedniego rozmiaru dla wszystkich członków klubu, czy też nie.*

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*
`4 4 2 1`	`TAK`
`1 3`	`TAK`
`2 3`	`NIE`
`3 3`	`TAK`
`2 -1`	

Po zajściu wszystkich zdarzeń z podanej sekwencji mamy trzech członków klubu, którzy mogą nosić łyżwy rozmiaru 1 lub 2, dwóch członków, którzy mogą nosić łyżwy rozmiaru 2 lub 3, oraz trzech, którzy mogą nosić łyżwy rozmiaru 3 lub 4. Przy takim składzie klubu rzeczywiście wystarczą po dwie pary łyżew rozmiarów 1, 2, 3 i 4:

- *dwie osoby dostają łyżwy rozmiaru 1;*
- *łyżwy rozmiaru 2 dostaje jedna osoba, która może nosić łyżwy rozmiaru 1 lub 2, i jedna, która może nosić łyżwy rozmiaru 2 lub 3;*
- *łyżwy rozmiaru 3 dostaje jedna osoba, która może nosić łyżwy rozmiaru 2 lub 3, i jedna, która może nosić łyżwy rozmiaru 3 lub 4;*
- *pozostałe dwie osoby dostają łyżwy rozmiaru 4.*

Rozwiązanie

Wersja statyczna

W przypadku problemów, w których mamy do czynienia z pewnym zbiorem obiektów zmieniających się w czasie i naszym zadaniem jest odpowiadanie na zapytania związane ze stanem tych obiektów w różnych momentach, zazwyczaj warto zadać sobie pytanie, jak należałoby odpowiadać na takie zapytania w przypadku obiektów, które nie zmieniają się w ogóle. Innymi słowy, możemy chcieć uprościć problem z wersji *dynamicznej*, czyli zmiennej w czasie, do *statycznej*. Spróbujemy zastosować to podejście do zadania o łyżwiarzach i łyżwach.

Ponieważ w zadaniu pojawia się stosunkowo dużo różnych danych i parametrów, nasz opis rozpoczniemy od krótkiego sformułowania problemu, który zamierzamy rozwiązać.

Problem w wersji statycznej.
Dane wejściowe:

- w problemie występują łyżwy o rozmiarach od 1 do n;
- łyżew rozmiaru j jest $s_j = k$ par;
- mamy t_1 łyżwiarzy o rozmiarze stopy 1, t_2 — o rozmiarze 2, itd., wreszcie t_{n-d} łyżwiarzy o rozmiarze stopy $n-d$;

- łyżwiarze z grupy t_i mogą nosić łyżwy rozmiarów z przedziału $[i, i+d]$.

Należy odpowiedzieć `TAK` lub `NIE`, w zależności od tego, czy można wszystkim łyżwiarzom przydzielić odpowiednie łyżwy.

Zauważmy, że jeżeli znajdziemy rozwiązanie problemu w wersji statycznej, to będziemy mogli je zastosować bezpośrednio w wersji dynamicznej przy każdym z m zapytań z osobna. Patrząc na ograniczenia z zadania, łatwo wywnioskować, że zapewne nie uzyskamy w ten sposób dostatecznie efektywnego rozwiązania, ale będziemy mieli w ogóle od czego zacząć myśleć dalej.

Rozwiązanie zachłanne

Okazuje się, że przy tak postawionym problemie, jeżeli istnieje żądane przyporządkowanie, to można je skonstruować, przydzielając łyżwy w sposób zachłanny. Dokładniej, analizujemy łyżwiarzy w kolejności $t_1, t_2, \ldots, t_{n-d}$ i każdemu z nich przypisujemy wolne łyżwy o najmniejszym możliwym rozmiarze.

Zanim zaczniemy się zastanawiać, jak efektywnie zaimplementować takie rozwiązanie (wszak łączne liczby łyżwiarzy i łyżew mogą być bardzo duże!), odpowiedzmy sobie na pytanie, dlaczego opisany algorytm zachłanny działa. Oczywiście można pominąć formalne dowodzenie i założyć, że „na pewno się nie pomyliliśmy", ale przy algorytmach zachłannych często warto być ostrożnym — przekonała się o tym m.in. liczna grupa zawodników, którzy w zadaniu *Konduktor* z tego samego etapu zaimplementowali „równie oczywiste" rozwiązanie zachłanne i skończyli z niedodatnią punktacją. A zatem:

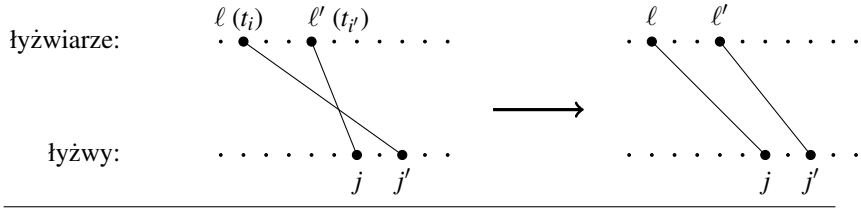
Twierdzenie 1. *Jeżeli problem w wersji statycznej posiada rozwiązanie, to opisany algorytm zachłanny znajdzie je.*

Dowód: Wystarczy pokazać, że jeżeli algorytm zachłanny udzieli odpowiedzi `NIE`, to w ogóle nie istnieje żadne poprawne przyporządkowanie łyżew łyżwiarzom (reguła kontrapozycji).

Dowód przeprowadzimy przez sprzeczność. Załóżmy, że algorytm zachłanny skonstruował jakieś przyporządkowanie cząstkowe C i następnie „zaciął" się na pewnym łyżwiarzu, któremu nie mógł już przypisać odpowiednich łyżew, istnieje natomiast pełne przyporządkowanie P, jednakże skonstruowane w jakiś inny sposób. Przyporządkowania C i P muszą się gdzieś różnić. Niech zatem i będzie najmniejszym takim rozmiarem stopy, że pewnemu łyżwiarzowi ℓ z grupy t_i w przyporządkowaniu C zostały przydzielone łyżwy rozmiaru j, natomiast w przyporządkowaniu P — o rozmiarze $j' \neq j$. Pokażemy, że możemy tak poprzestawiać przypisania w ramach P, żeby nie zmienić łyżew żadnego z wcześniejszych łyżwiarzy i przypisać łyżwiarzowi ℓ łyżwy rozmiaru j. Stąd uzyskamy żądaną sprzeczność, albowiem kontynuując tego typu przekształcanie przyporządkowania P, uzyskamy dokładnie taki przydział, jak w przyporządkowaniu C, co kłóci się z tym, że C nie da się już powiększyć o kolejne przypisanie.

Opiszemy teraz, jak wykonać zapowiadane przekształcenie. Zauważmy, że skoro C było konstruowane w sposób zachłanny, to $j < j'$. Zapytajmy się zatem, kto w przyporządkowaniu P jest właścicielem łyżew rozmiaru j. Ponieważ dla wszystkich wcześniejszych łyżwiarzy P i C są zgodne, więc w P jakieś łyżwy rozmiaru j są albo w ogóle nieprzypisane, albo znajdują się na nogach jakiegoś łyżwiarza ℓ' o rozmiarze stopy $i' \geqslant i$. W pierwszym z tych przypadków

opisane przekształcenie jest bardzo proste: zdejmujemy łyżwiarzowi ℓ jego aktualne łyżwy, a zakładamy wolne łyżwy rozmiaru j. W przeciwnym przypadku też nie będzie zbyt ciężko: wystarczy zamienić łyżwy łyżwiarzom ℓ i ℓ'. Jest to możliwe, gdyż przypisania łyżew tym łyżwiarzom „krzyżują się" (patrz rys. 1), tj. $i \leqslant i' \leqslant j \leqslant j'$. Skoro więc $j' \leqslant i + d$, to tym bardziej $j' \leqslant i' + d$ oraz $j \leqslant i + d$ i faktycznie zamiana łyżew jest możliwa. Jako że rozważyliśmy oba przypadki i w każdym wskazaliśmy żądane przekształcenie, to dowód jest zakończony. ■



Rys. 1: Zamiana łyżew (w ramach P) między łyżwiarzami ℓ oraz ℓ' o rozmiarze stopy odpowiednio i oraz i'.

Dodajmy, że powyższe uzasadnienie poprawności podejścia zachłannego jest dosyć typowe — warto zapamiętać tę metodę dowodzenia, gdyż może się ona przydać w przypadku innych zadań.

Implementacja rozwiązania zachłannego

Przyszła pora na zmierzenie się z implementacją algorytmu zachłannego. Ze względu na wspomniane już duże liczby łyżwiarzy i łyżew, przyporządkowań będziemy dokonywać „hurtowo", w jednym kroku obsługując wszystkich łyżwiarzy z danej grupy t_i.

```
1:  Algorytm zachłanny – implementacja 1
2:    for i := 1 to n − d do
3:    begin
4:      for j := i to i + d do
5:      begin
6:        g := min(t_i, s_j);
7:        t_i := t_i − g;
8:        s_j := s_j − g;
9:      end
10:     if t_i > 0 then return NIE;
11:   end
12:   return TAK;
```

Złożoność czasowa takiego rozwiązania to ewidentnie $O(nd)$. Okazuje się jednak, że w powyższej implementacji często nadrabiamy pracy, chociażby w sytuacji, gdy wielokrotnie próbujemy przydzielić łyżwiarzom łyżwy rozmiaru, który już dawno został wyczerpany. Można temu jednakże zaradzić. Zauważmy mianowicie, że w każdym kroku wewnętrznej pętli **for** jedna ze zmiennych t_i, s_j zostaje wyzerowana. Jeżeli jest to t_i, to możemy w ogóle wyjść z tej pętli. Jeżeli natomiast s_j, to wiemy, że łyżew rozmiaru j już nigdy więcej

nie musimy rozważać. Korzystając z tej obserwacji, możemy sprytniej zapisać powyższy pseudokod:

```
1:  Algorytm zachłanny – implementacja 2
2:    j := 1;
3:    for i := 1 to n − d do
4:    begin
5:      if j < i then j := i;
6:      while t_i > 0 and j ⩽ i + d do
7:      begin
8:        g := min(t_i, s_j);
9:        t_i := t_i − g;
10:       s_j := s_j − g;
11:       if s_j = 0 then j := j + 1;
12:     end
13:     if t_i > 0 then return NIE;
14:   end
15:   return TAK;
```

Jaka jest złożoność czasowa powyższego rozwiązania? Co prawda w pseudokodzie występują dwie zagnieżdżone pętle, jednak łączna liczba obrotów tych pętli jest niewielka, a to za sprawą tego, że w każdym kroku pętli **while** wartość zmiennej j wzrasta o jeden (jeśli zachodzi warunek w 11. linii), bądź też jest to ostatnie wykonanie wewnątrz danego obrotu pętli **for** (gdyż $t_i = 0$). To pokazuje, że pętla **while** wykona łącznie co najwyżej $2n$ obrotów, więc złożoność czasowa tej implementacji algorytmu to właśnie $O(n)$. Uzyskaliśmy zatem istotne przyspieszenie algorytmu za pomocą zaledwie drobnych modyfikacji pseudokodu.

Rozwiązania wersji dynamicznej problemu używające pierwszej z powyższych implementacji można znaleźć w plikach: `lyzs1.cpp`, `lyzs2.pas` i `lyzs3.java`. Mają one złożoność czasową $O(mnd)$ i zdobywały na zawodach 10% punktów. W przypadku drugiej z powyższych implementacji otrzymujemy rozwiązanie o złożoności $O(mn)$, które zdobywało na zawodach 30% punktów. Implementacje w plikach: `lyzs4.cpp`, `lyzs5.pas` i `lyzs6.java`.

Problem w wersji dynamicznej można rozwiązać efektywniej, ale nie tędy droga. Otóż niestety rozwiązanie zachłanne nie bardzo nadaje się do dalszych ulepszeń. W następnej sekcji zajmiemy się opisem innego rozwiązania problemu statycznego, co prawda o tej samej złożoności czasowej, ale efektywniejszego pod kątem wersji dynamicznej. Na szczęście praca nad rozwiązaniem zachłannym nie pójdzie na marne: fakt, że jest ono poprawne (Twierdzenie 1), przyda nam się do dowodu poprawności tego nowego rozwiązania.

Inne rozwiązanie wersji statycznej

Tym razem do rozwiązania podejdziemy zupełnie inaczej: wyznaczymy pewne zwięźle formułowalne kryterium na to, kiedy da się przydzielić wszystkim łyżwiarzom pasujące łyżwy, a kiedy nie.

Przyjrzyjmy się łyżwiarzom o rozmiarach stopy z przedziału $[a,b]$. Jasne jest, że wszystkie rozmiary łyżew, które mają szansę pasować tym łyżwiarzom, należą do przedziału $[a,b+d]$. Aby istniała w ogóle możliwość odpowiedzi pozytywnej w naszym problemie,

liczba par łyżew o rozmiarach z przedziału $[a, b+d]$ musi być nie mniejsza niż łączna liczba łyżwiarzy $t_a + t_{a+1} + \ldots + t_b$. Jest to zatem pewien warunek konieczny istnienia poprawnego przyporządkowania łyżew. Co ciekawe, warunek ten zapisany w odpowiedni sposób staje się także warunkiem dostatecznym! Mówi o tym następujące twierdzenie.

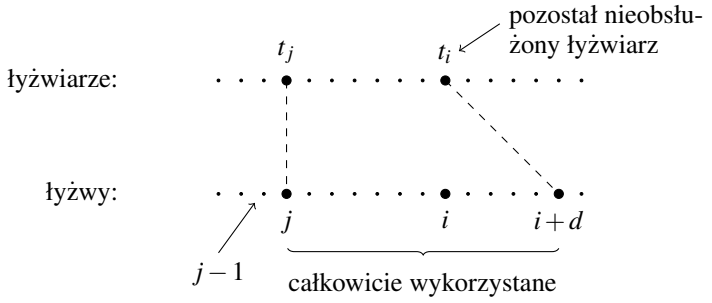
Twierdzenie 2. *Przydział łyżew jest możliwy wtedy i tylko wtedy, gdy dla każdego przedziału* $[a,b]$*, takiego że* $1 \leqslant a \leqslant b \leqslant n-d$*, zachodzi*

$$t_a + t_{a+1} + \ldots + t_b \leqslant (b+d-a+1) \cdot k. \tag{1}$$

Dowód: Uzasadniliśmy już, że każdy z warunków (1) z osobna stanowi warunek konieczny istnienia przydziału łyżew. Ograniczymy się zatem do pokazania dostateczności układu wszystkich warunków (1), tzn. tego, że jeżeli wszystkie warunki zachodzą, to istnieje żądane przyporządkowanie łyżew łyżwiarzom. To sformułowanie jest z kolei równoważne temu, że jeżeli takie przyporządkowanie nie istnieje, to któryś z warunków (1) nie zachodzi.

Załóżmy zatem, że nie istnieje sposób przydziału łyżwiarzom odpowiednich łyżew. Na mocy Twierdzenia 1 oznacza to, że dla takich danych wejściowych algorytm zachłanny nie znajdzie pełnego przyporządkowania. Korzystając z tego spostrzeżenia, wskażemy przedział $[a,b]$, dla którego kryterium (1) nie zachodzi.

Porażka algorytmu zachłannego polega na tym, że pewnemu łyżwiarzowi z grupy t_i nie udało się przyporządkować żadnych łyżew, gdyż wszystkie łyżwy o rozmiarach od i do $i+d$ zostały już wcześniej wykorzystane. Oznaczmy przez C znalezione przyporządkowanie częściowe. Niech j będzie najmniejszym rozmiarem łyżew takim, że wszystkie łyżwy o rozmiarach $j, j+1, \ldots, i+d$ zostały wykorzystane przy konstruowaniu C. Wykażemy, że szukanym przedziałem $[a,b]$ jest przedział $[j,i]$ (patrz rys. 2).



Rys. 2: W przyporządkowaniu częściowym C łyżwy o rozmiarach z przedziału $[j, i+d]$ zostały całkowicie wykorzystane (po k par), natomiast łyżwy rozmiaru $j-1$ już nie. To oznacza, że tylko łyżwiarze z grup $t_j, \ldots, t_i$ mają na nogach łyżwy o rozmiarach z przedziału $[j, i+d]$ i, co więcej, jeszcze tych łyżew dla nich zabrakło.

W tym celu wystarczy stwierdzić, że żaden z łyżwiarzy z grup $t_1, t_2, \ldots, t_{j-1}$ nie dostał w ramach C łyżew rozmiaru j lub większego. Jeżeli $j = 1$, to to zdanie jest oczywiście prawdziwe. W przeciwnym przypadku na mocy definicji j wiemy, że w ramach C pozostały niewykorzystane jakieś łyżwy rozmiaru $j-1$. A ponieważ C zostało skonstruowane w sposób zachłanny, więc żadnemu łyżwiarzowi z grupy t_q, $q < j$, nie mogły zostać przydzielone łyżwy

o rozmiarze nie mniejszym niż j, gdyż wcześniej otrzymałby jedną z wolnych par łyżew rozmiaru $j-1$.

Z dotychczasowych spostrzeżeń wynika, że wszystkie $(i+d-j+1)\cdot k$ łyżew o rozmiarach od j do $i+d$ zostało wykorzystanych oraz że łyżwy te zostały rozdzielone wśród $t_j+t_{j+1}+\ldots+t_i$ łyżwiarzy o rozmiarach stopy między j a i, i na dodatek jeszcze ich zabrakło. To pokazuje, że rzeczywiście kryterium nie zachodzi w przypadku $a=j$, $b=i$. ■

Ktoś mógłby zapytać, skąd właściwie wzięło się takie „magiczne" kryterium istnienia przyporządkowania i w jaki sposób można na nie wpaść. Jednym ze sposobów jest na pewno metoda prób i błędów — zauważamy konieczność układu warunków (1) i mimo usilnych starań nie jesteśmy w stanie wymyślić kontrprzykładu na ich dostateczność. Okazuje się, że istnieje także bardziej metodyczne podejście do problemu, które pozwala wywnioskować Twierdzenie 2 jako szczególny przypadek pewnej własności grafowej. Co ciekawe, sprowadzenie to zawdzięczamy samym zawodnikom — dziwnym trafem żaden z jurorów nie wykrył go przed zawodami.

Zauważmy mianowicie, że łyżwy i łyżwiarze tworzą pewien graf dwudzielny $G=(V_1,V_2,E)$, o czym zresztą mogły nas już przekonać dotychczasowe rysunki w opisie rozwiązania. Łyżwiarz $v_1\in V_1$ jest w nim połączony krawędzią z parą łyżew $v_2\in V_2$ wtedy i tylko wtedy, gdy może w ramach swojego przedziału tolerancji założyć te łyżwy. Szukane przyporządkowanie łyżew jest wówczas *skojarzeniem* w grafie G (czyli takim podzbiorem krawędzi, z których żadne dwie nie są incydentne), w którym *każdy* łyżwiarz jest skojarzony z jakąś parą łyżew. Kryterium istnienia takiego skojarzenia stanowi klasyczne Twierdzenie Halla (patrz [32]):

Twierdzenie 3 (Hall). *W grafie dwudzielnym $G=(V_1,V_2,E)$ istnieje skojarzenie pokrywające wszystkie wierzchołki ze zbioru V_1 wtedy i tylko wtedy, gdy dla każdego podzbioru $W_1\subseteq V_1$ moc zbioru wierzchołków sąsiadujących z jakimikolwiek wierzchołkami z W_1:*

$$W_2=\{v_2\in V_2:\ \exists_{v_1\in W_1}\ (v_1,v_2)\in E\}$$

jest nie mniejsza od mocy W_1, czyli $|W_2|\geqslant|W_1|$.

Aby z Twierdzenia Halla uzyskać Twierdzenie 2, wystarczy udowodnić:

Lemat 1. W przypadku grafu łyżwiarzy i łyżew warunek istnienia skojarzenia z Twierdzenia Halla jest równoważny warunkowi (1).

Dowód: Zauważmy, że warunek (1), jeżeli wyrazić go w języku teorii grafów, stanowi szczególny przypadek warunku z twierdzenia Halla, w którym zbiór W_1 składa się ze wszystkich łyżwiarzy o rozmiarach stopy od a do b. Teza lematu w jedną stronę jest więc oczywista.

W drugą stronę musimy pokazać, że warunek (1) implikuje warunek z Twierdzenia Halla. W tym celu wystarczy udowodnić, że jeżeli warunek $|W_2|\geqslant|W_1|$ nie zachodzi dla pewnego $W_1\subseteq V_1$, to istnieje także jakiś zbiór W_1' składający się ze wszystkich łyżwiarzy o rozmiarach stopy z pewnego przedziału $[a,b]$, dla którego nie zachodzi analogiczny warunek $|W_2'|\geqslant|W_1'|$.

Niech W_1 będzie najmniej licznym zbiorem, dla którego warunek z twierdzenia Halla nie zachodzi. Zauważmy, że wówczas nie może istnieć w W_1 para kolejnych pod względem rozmiaru stopy łyżwiarzy, których rozmiary stopy różnią się o więcej niż d, gdyż wówczas

wyznaczaliby oni podział W_1 na dwa mniejsze zbiory, z których dla co najmniej jednego nie zachodziłoby kryterium Halla. Niech l i u oznaczają odpowiednio najmniejszy i największy rozmiar stopy łyżwiarza w zbiorze W_1. Wówczas W_2 składa się ze wszystkich dostępnych par łyżew o rozmiarach z przedziału $[l, u+d]$. Ponadto, dołożenie do W_1 wszystkich łyżwiarzy o rozmiarach stopy od l do u nie zmienia postaci odpowiadającego zbioru W_2. W wyniku takiego powiększenia W_1 uzyskujemy szukany zbiór W_1' odpowiadający przedziałowi rozmiarów stopy $[a,b] = [l,u]$. ■

W ten sposób uzyskaliśmy alternatywne wyprowadzenie kryterium z Twierdzenia 2. Godne podkreślenia jest, że wykorzystaliśmy do tego celu twierdzenie, które na pierwszy rzut oka wydaje się być zupełnie bezużyteczne w praktyce — istnieją wszakże wielomianowe algorytmy znajdowania najliczniejszego skojarzenia w grafie dwudzielnym (patrz np. [20]), a przecież Twierdzenie Halla sprowadza problem skojarzenia do sprawdzenia wykładniczej liczby warunków! Siła tego twierdzenia ujawniła się w tym przypadku w związku ze szczególną postacią naszego grafu dwudzielnego — podobne przykłady można znaleźć w literaturze (patrz np. [32]).

Podciąg spójny o maksymalnej sumie

Aby skutecznie wykorzystać kryterium z Twierdzenia 2, trzeba trochę je przeformułować. Warunek (1) możemy zapisać równoważnie tak:

$$(t_a - k) + (t_{a+1} - k) + \ldots + (t_b - k) \leqslant d \cdot k.$$

Wprowadzając nowe zmienne $t'_1, t'_2, \ldots, t'_{n-d}$, tak aby zachodziła równość $t'_i = t_i - k$, otrzymujemy następujący zapis wyprowadzonego kryterium:

Twierdzenie 4. *Przydział łyżew jest możliwy wtedy i tylko wtedy, gdy dla każdego przedziału $[a,b]$, takiego że $1 \leqslant a \leqslant b \leqslant n-d$, zachodzi*

$$t'_a + t'_{a+1} + \ldots + t'_b \leqslant d \cdot k. \tag{2}$$

Jaką korzyść uzyskaliśmy z takiego przekształcenia? Zauważmy jedną istotną rzecz: po prawej stronie każdego z warunków (2) występuje wyrażenie niezależne od parametrów a i b, to znaczy *stałe* dla ustalonych danych wejściowych. Skoro tak, to możemy podejść do sprawdzenia warunku z nowego kryterium w trochę nietypowy sposób. Otóż wystarczy znaleźć takie a i b, dla których wartość wyrażenia $t'_a + t'_{a+1} + \ldots + t'_b$ jest *największa*, i sprawdzić, czy ta właśnie wartość przekracza $d \cdot k$, czy nie.

W ten sposób wyjściowy problem został sprowadzony do klasycznego problemu podciągu spójnego ciągu t'_i o maksymalnej sumie[1]. Znane są co najmniej dwa algorytmy rozwiązujące ten problem (patrz [19]) w złożoności czasowej liniowej względem długości ciągu, czyli w tym przypadku $O(n)$. Implementacje rozwiązania problemu dynamicznego opartego o tę metodę można znaleźć w plikach `lyzs7.cpp`, `lyzs8.pas` i `lyzs9.java`. Mają one złożoność czasową $O(mn)$ i zdobywały na zawodach 30% punktów.

[1] Warto zaznaczyć, że elementy ciągu t'_i, w przeciwieństwie do t_i, nie muszą być nieujemne.

Wersja dynamiczna

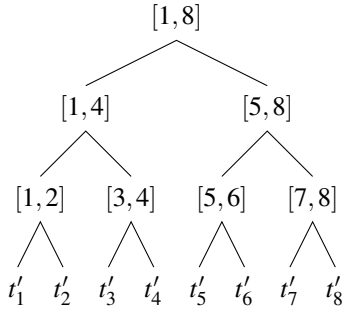
Podobnie jak w przypadku statycznym, na początku sformułujemy dokładnie problem, który chcemy rozwiązać. Zauważmy, że dzięki sprowadzeniu z poprzedniej sekcji możemy pominąć wiele parametrów problemu i opisać go abstrakcyjnie. W dalszym opisie będziemy stosowali skrótowe oznaczenie *psoms* = podciąg spójny o maksymalnej sumie.

Dynamiczny problem psoms.
Dany jest ciąg $t'_i: \ i = 1,2,\ldots,n-d$, początkowo $t'_i \equiv -k$. Należy zaprojektować strukturę danych, która udostępnia następujące operacje:

- `modyfikuj`(i, x): zwiększenie t'_i o stałą x (x może też być ujemne);
- `zapytanie`: wyznaczenie sumy psoms w ciągu t'_i.

Naszym celem jest jak najefektywniejsze przeprowadzenie inicjalizacji tej struktury (operacja `init`) oraz zasymulowanie m par podanych operacji.

Poszukiwaną strukturą danych może być na przykład statyczne drzewo przedziałowe[2], będące pełnym drzewem binarnym, w którego liściach umieszczone są elementy ciągu t'_i, a węzły odpowiadają przedziałom indeksów tego ciągu. Dla uproszczenia zakładamy, że liczba elementów ciągu t'_i jest całkowitą potęgą dwójki (jeżeli tak nie jest, to możemy zawsze dołożyć jakieś sztuczne elementy o wartości 0). Przykład takiego drzewa można znaleźć na rys. 3.



Rys. 3: Przykład drzewa przedziałowego dla ośmioelementowego ciągu t'_i.

Podstawowe własności drzewa przedziałowego, które będą nam potrzebne, to:

- liczba węzłów drzewa jest liniowa względem długości ciągu, a głębokość drzewa — logarytmiczna;
- ponumerowane w porządku kopcowym drzewo możemy reprezentować w zwykłej tablicy;

[2]Więcej o statycznych drzewach przedziałowych można przeczytać np. w opisie rozwiązania zadania Tetris 3D z książeczki XIII Olimpiady Informatycznej [13], albo poczytać i posłuchać na stronie `http://was.zaa.mimuw.edu.pl`

- przedziały zawarte w dzieciach węzła odpowiadają połowicznemu podziałowi przedziału tego węzła pod względem zbioru liczb całkowitych w nim zawartych;
- wszystkie przedziały zawierające dany indeks i stanowią ścieżkę od liścia odpowiadającego t'_i do korzenia.

Aby rozwiązać dynamiczny problem psoms, wzbogacimy drzewo przedziałowe, umieszczając we wszystkich węzłach pewne dodatkowe wartości, które pomogą nam efektywnie odpowiadać na zapytania. Zaczniemy od wprowadzenia pola `max_psoms`, w którym będziemy przechowywać maksymalną sumę psoms podciągu odpowiadającego danemu węzłowi (jeżeli wszystkie wyrazy tego podciągu są ujemne, to prawidłową wartością tego pola będzie 0). Jeżeli będziemy umieli utrzymywać te wartości, symulując operację `modyfikuj`, to obsługa zapytania będzie polegała zaledwie na odczytaniu wartości `max_psoms` z korzenia drzewa.

Aby dało się aktualizować wartości opisanego parametru podczas operacji modyfikacji, wystarczy zadbać o to, aby wartość `max_psoms` w węźle p zależała jedynie od wartości dla synów l i r tego węzła oraz ewentualnie pewnych dodatkowych pól, które jeszcze trzeba będzie wprowadzić. Jeżeli uda nam się to zagwarantować, to modyfikacja będzie polegała na zmianie wartości t'_i w odpowiednim liściu oraz na poprawieniu wartości pól na ścieżce od tego liścia do korzenia drzewa.

Jak zatem obliczać wartość `max_psoms`(p)? Na pewno zależy ona od maksimum z wartości `max_psoms`(l) oraz `max_psoms`(r) — dotyczy to przypadku, kiedy psoms w p jest w całości zawarty w jednej z połówek przedziału — ale jeżeli psoms zawiera zarówno pewne elementy z l, jak i z r, to do wyznaczenia tej wartości potrzebne jest coś jeszcze. W tym przypadku psoms dla p składa się z pewnego *sufiksu* (tj. końcowego fragmentu) podciągu odpowiadającego l oraz z pewnego *prefiksu* (tj. początkowego fragmentu) podciągu r. A jakiego sufiksu i prefiksu? Oczywiście tych o największych sumach!

W ten sposób wywnioskowaliśmy potrzebę wprowadzenia pól `max_pref` oraz `max_suf` dla węzłów. Po chwili namysłu można dalej wywnioskować, że z kolei do wyznaczania tych wartości potrzebne jest jeszcze pole `suma`, oznaczające sumę wszystkich elementów podciągu odpowiadającego danemu węzłowi drzewa.

Podsumujmy ten luźny wywód dokładnymi definicjami wprowadzonych pól dla węzła odpowiadającego przedziałowi $[a,b]$:

- `max_psoms`: suma psoms ciągu $t'_a, t'_{a+1}, \ldots, t'_b$;
- `max_pref`: maksymalna z sum prefiksowych postaci $t'_a + t'_{a+1} + \ldots + t'_i$ dla $i \leqslant b$;
- `max_suf`: maksymalna z sum sufiksowych postaci $t'_i + t'_{i+1} + \ldots + t'_b$ dla $i \geqslant a$;
- `suma`: $t'_a + t'_{a+1} + \ldots + t'_b$;

oraz odpowiadającymi im wzorami rekurencyjnymi:

$$\begin{aligned}
\texttt{max_psoms}(p) &= \max(\texttt{max_psoms}(l),\ \texttt{max_psoms}(r), \\
&\qquad \texttt{max_suf}(l) + \texttt{max_pref}(r)) && (3)\\
\texttt{max_pref}(p) &= \max(\texttt{max_pref}(l),\ \texttt{suma}(l) + \texttt{max_pref}(r)) && (4)\\
\texttt{max_suf}(p) &= \max(\texttt{max_suf}(r),\ \texttt{max_suf}(l) + \texttt{suma}(r)) && (5)\\
\texttt{suma}(p) &= \texttt{suma}(l) + \texttt{suma}(r) && (6)
\end{aligned}$$

Dodajmy, że w liściu l odpowiadającemu elementowi t'_i mamy:

$$\texttt{max\_psoms}(l) = \texttt{max\_pref}(l) = \texttt{max\_suf}(l) = \max(t'_i, 0) \tag{7}$$

$$\texttt{suma}(l) = t'_i \tag{8}$$

Za pomocą podanych wzorów możemy już zaimplementować żądane operacje:

- `init`: za pomocą (7) i (8) inicjujemy wartości dla liści, po czym za pomocą wzorów (3)–(6) dla kolejnych węzłów wewnętrznych (warstwami od dołu). Złożoność czasowa $O(n)$.
- `modyfikuj`(i, x): korzystając ze wzorów (7) i (8), ustawiamy wartości w liściu l odpowiadającemu t'_i, po czym aktualizujemy wszystkie wartości na ścieżce od l do korzenia (wzory (3)–(6)). Złożoność czasowa pojedynczej operacji: $O(\log n)$.
- `zapytanie`: zwracamy `max_psoms` dla korzenia (czas $O(1)$).

Całkowity koszt czasowy inicjalizacji i wykonania m zapytań to $O(n + m \log n)$. Opisane rozwiązanie wzorcowe zostało zaimplementowane w plikach `lyz.cpp`, `lyz1.pas` oraz `lyz2.java`.

Testy

Rozwiązania zawodników były sprawdzane na 10 zestawach danych testowych. Sposób ich generowania był jednolity i w znacznej mierze losowy. Polegał on na powtarzaniu następujących czynności:

- wylosowanie przedziału $[a,b]$ rozmiarów stopy;
- dodawanie do klubu łyżwiarzy, których rozmiar stopy należy do przedziału $[a,b]$, aż do momentu, gdy nie można przyporządkować łyżew wszystkim łyżwiarzom (liczby dodawanych łyżwiarzy były tak dobierane, aby Bajtazar musiał korzystać z możliwie szerokiego zakresu rozmiarów łyżew przy rozdawaniu ich łyżwiarzom);
- usuwanie z klubu łyżwiarzy z przedziału rozmiarów stopy, który powoduje niezachodzenie kryterium (2);

do czasu, aż wykorzystany zostanie limit operacji. Poniższa tabelka zawiera parametry kolejnych testów, w kolejności: liczbę rozmiarów stopy, liczbę operacji, zapas łyżew każdego rozmiaru oraz przedział tolerancji stóp łyżwiarzy.

Nazwa	n	m	k	d	Opis
lyz1.in	500	1 000	12	10	mały test losowy, małe d
lyz2.in	5 000	10 000	54	2 000	mały test losowy, duże d
lyz3.in	5 000	10 000	72	2 000	mały test losowy, duże d
lyz4.in	15 000	30 000	6 740	70	średni test losowy

Nazwa	n	m	k	d	Opis
lyz5.in	100 000	250 000	57 434	30	duży test losowy
lyz6.in	100 000	250 000	23 522	40	duży test losowy
lyz7.in	200 000	500 000	1 012	50	duży test losowy
lyz8.in	200 000	500 000	512 300 012	21 212	duży test losowy
lyz9.in	200 000	500 000	999 999 997	23 500	duży test losowy
lyz10a.in	200 000	500 000	1 000 000 000	50 000	duży test losowy
lyz10b.in	200 000	500 000	1 000 000 000	0	duży test losowy z zerową tolerancją

Zadanie dodatkowe

Jeżeli spodobało Ci się, drogi Czytelniku, zadanie o łyżwiarzach i łyżwach, to możesz zechcieć zmierzyć się z innym, nieco podobnym problemem. Dane są w nim górne ograniczenia $a_1, a_2, \ldots, a_n$ na wartości elementów permutacji zbioru $\{1, 2, \ldots, n\}$ i należy odpowiadać na zapytania o to, czy istnieje jakakolwiek permutacja spełniająca te ograniczenia. Dodatkowo, ograniczenia te zmieniają się (pojedynczo) w czasie. Zainteresowanych Czytelników odsyłamy do zadania *Permutacja* z piątej rundy Potyczek Algorytmicznych 2009.

Zawody III stopnia

opracowania zadań