Zadanie 2. *Danych jest n odcinków $I_j = (p_j, k_j)$, leżących na osi OX, $j = 1, ..., n$. Ułóż algorytm znajdujący zbiór $S \subseteq \{I_1, ..., I_n\}$ nieprzecinających się odcinków o największej mocy.*

Rozwiązanie

Najpierw intuicja: wzięcie takiego elementu I_j, który ma minimalne k_j, jest zawsze dobre. Wszystkie odcinki, które się z nim przecinają, mają końce nie bliżej niż I_j, a więc parami też się przecinają, bo wszystkie zawierają punkt k_j. Stąd możemy wziąć tylko jeden z tych odcinków. Dodatkowo wziąć najwcześniejszy koniec nam się opłaca, aby potencjalnie uniknąć przecięcia się z „późniejszymi” odcinkami, z poza tego przecinającego się zbioru wygenerowanego z I_j.

Spróbujmy uogólnić tę intuicję na cały zbiór odcinków $I = \{I_1, ..., I_n\}$ w ten sposób: najpierw posortujmy rosnąco I po końcach, czyli drugim elemencie pary, umieśćmy pierwszy w kolejności element w zbiorze wynikowym S, a potem dołączajmy następne, pod warunkiem, że nie przecinają się z żadnym już uwzględnionym.

Ostatni warunek możemy sprawdzać przechowując koniec ostatniego dołączonego odcinka. Jeżeli rozważany odcinek zaczyna się wcześniej, niż ten koniec, to odcinki się przecinają. W ten sposób ze względu na konieczność sortowania uzyskujemy algorytm o złożoności $O(n \log n)$.

Procedure MaxDisjointSet(I)

$I' \leftarrow$ SortBySnd(I) — zbiór odcinków posortowany po końcach
$S \leftarrow \{I'_1\}$ — zbiór przechowujący wynik
$r \leftarrow k'_1$ — ostatni koniec, jaki widzieliśmy
for $i \leftarrow 2$ **to** n **do**
    if $p'_i > r$ **then**
        $S \leftarrow S \cup \{I'_i\}$
        $r \leftarrow k'_i$
    end
end
return S

Poprawność

Sprawdźmy, że zaprezentowany algorytm zachłanny jest poprawny: zawsze wylicza zbiór rozłącznych odcinków o największej mocy.

Zdefiniujmy precyzyjny porządek na odcinkach.

$$I_i \preceq I_j \longleftrightarrow k_i < k_j \vee (k_i = k_j \wedge p_i \leqslant p_j)$$

Będziemy zakładać, że jest to porządek stosowany przez algorytm w procedurze SortBySnd, choć to założenie nie jest konieczne dla poprawności algorytmu.

Dodatkowo wprowadźmy następujące oznaczenie.

$$I_i \prec I_j \longleftrightarrow I_i \preceq I_j \land I_i \neq I_j$$

Twierdzenie 1. *Zbiór wyliczony przez algorytm zawsze składa się z parami rozłącznych odcinków.*

Dowód. Niech I będzie dowolnym zbiorem odcinków, a $S = \{I_1, I_2, ..., I_k\}$ to odcinki wybrane przez algorytm (w kolejności dodawania ich do zbioru) dla tego zbioru.

Oczywiście $I_1 \preceq I_2 \preceq ... \preceq I_k$, bo w takim porządku przeglądamy kolejne elementy I.

Rozważmy dowolną parę odcinków taką, że $I_i \preceq I_j$. Podczas rozważania odcinka I_i zmienna r w linii 7. została ustawiona na k_i. Wszystkie kolejne elementy w I' miały końce nie mniejsze niż k_i, więc zmienna r już nigdy nie zmalała. I_j został dodany do zbioru S, a więc spełnia warunek $p_j > r$. Skoro był dodany później niż I_i, to spełnia także $p_j > r_i$. Zatem odcinki są rozłączne. □

Twierdzenie 2. *Zbiór wyliczony przez algorytm jest optymalny.*

Dowód. Niech I będzie dowolnym zbiorem odcinków. Niech $S = \{I_1, I_2, ..., I_k\}$ będzie zbiorem wyznaczonym przez nasz algorytm. Załóżmy nie wprost, że istnieje większy zbiór. Niech $S' = \{J_1, J_2, ..., J_m\}$ będzie dowolnym zbiorem parami rozłącznych odcinków z I takim, że $m > k$. Dodatkowo załóżmy, że I_i i J_i mają zachowany porządek $\preceq$.

Lemat 1. *Dla dowolnego $i \in \{1, 2, ..., k\}$ zachodzi $I_i \preceq J_i$.*

Dowód. Indukcja względem k.

Podstawa indukcji. Rozważmy $k = 1$. W algorytmie wybraliśmy I_1 minimalne ze względu na porządek $\preceq$, więc $I_1 \preceq J_1$.

Krok indukcyjny. Załóżmy, że dla k teza zachodzi. Rozważmy $k+1$. Wiemy, że $I_k \preceq J_k$ oraz $J_k \prec J_{k+1}$, a więc $I_k \prec J_{k+1}$.

Dodatkowo $k(I_k) \leqslant k(J_k)$ (z def. $\preceq$) i $k(J_k) < p(J_{k+1})$ (bo są rozłączne). Zatem $k(I_k) < p(J_{k+1})$, więc te dwa odcinki także są rozłączne.

Algorytm wybiera I_{k+1} najmniejsze spośród nierozważanych, które jest rozłączne z już wybranymi odcinkami. J_{k+1} nie było rozważane przez algorytm przed $k+1$ (bo $I_k \prec J_{k+1}$). Więc skoro I_{k+1} to minimum, to $I_{k+1} \preceq J_{k+1}$. □

Z lematu wiemy, że $I_k \preceq J_k$. Stąd również $I_k \preceq J_{k+1}$. To oznacza, że nasz algorytm rozważa J_{k+1} później. Skoro J_{k+1} nie trafiło do S (bo I_k było ostatnie dołączone), to musi zachodzić $p(J_{k+1}) \leqslant k(I_k)$. Ale $k(I_k) \leqslant k(J_k)$, więc $p(J_{k+1}) \leqslant k(J_k)$. Z założenia $J_k \preceq J_{k+1}$, więc musi zachodzić $k(J_k) = k(J_{k+1})$ — czyli odcinki się przecinają. Sprzeczność. □

Z twierdzeń 1. i 2. wynika, że algorytm jest poprawny.