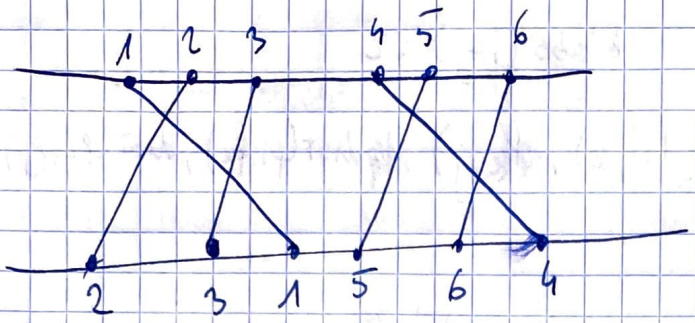
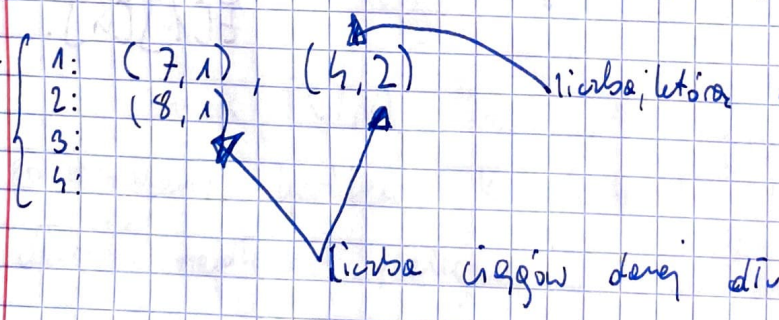
Z8|




Zauważmy że jeśli posortujemy punkty po x na jednej prostej i nadamy im odpowiadające numery na drugiej prostej to problem sprowadza się do znalezienia najdłuższego rosnącego podciągu punktów na drugiej prostej. (LIS)
Ten krok kosztuje nas $O(n \log n)$

b) ~~Dla~~ Przedstawię algorytm wyliczający liczbę LIS-ów ($O(n \log n)$) i pokażę jak wydobyć z niego wynik do a)

Tworzymy sobie strukturę, w której będziemy trzymać informacje ~~w $O(\log n)$~~ o wszystkich końcach podciągów danej długości:




- rozważając kolejną liczbę b.m.-searchem znajdujemy pierwszy taki wiersz, który kończy się liczbą < kandydat
- ustawiamy tam kandydata oraz korzystając z sum prefiksowych wyliczamy dla niego liczbę ciągów danej długości

suma -> suma[i-1].last - suma[i-1].pierwszy_większy + suma[i].zostaw_lewo

• na końcu w ostatnim wierszu sumę ~~[illegible]~~ ostatniej pary to interesującą nas wynik

a) Aby znaleźć ~~[illegible]~~ wystarczy się cofać z góry na dół
poprzedni
LIS
i brać pierwszy mniejszy element niż ostatni

Pseudokod do b):

```
vector < vector <int> > struktura;
max_i = 0
for number in numbers:
    i ← bin_search (number)          // znajdujemy największy taki wiersz, że
                                     number > ostatni w i-1
    max_i = max(i, max_i)
    sum ← struktura[i-1].back()[1] - bin_search2(i-1, number) + struktura[i].back()[1] // sumy prefiksowe

    struktura[i].push_back({number, sum})

return struktura[max_i].back()[1]
```

Z8

Możemy w tym problemie zauważyć, że można go sprowadzić do mniejszych podproblemów.

Jeśli ponumerujemy sobie wierzchołki od 1 do n, oraz wybierzemy punkt $p_k$ t.że $k \in \langle 2, n-1 \rangle \wedge k \in \mathbb{Z}$ to możemy podzielić wielokąt na: wielokąt $p_1 - p_k$, wielokąt $p_k - p_n$ oraz trójkąt $p_1, p_k, p_n$. Gdybyśmy mieli odpowiedzi dla wielokątów $p_1 - p_k$ oraz $p_k - n$. (Od teraz $P(i,j) \in$ odp dla $p_i - p_j$) to moglibyśmy wyliczyć rozwiązanie dla całego wielokąta:

$$\max\left(P(1,k), P(k,n), dist(p_1, p_k), dist(p_k, p_n)\right)$$, wtedy po rozważeniu każdego $k \in \langle 2, n-1 \rangle \wedge k \in \mathbb{Z}$ otrzymalibyśmy długość najdłuższej przekątnej przy triangulacji całego wielokąta.

Zauważmy, że aby policzyć $P(1,k) : P(k,n)$, możemy zastosować to samo rozumowanie powyżej otrzymujemy dwa wielokąty wypukłe, na których możemy rozważać nowe punkty $p_k$.

Czyli w sumie musimy wyliczyć $O(n^2)$ wartości $P(i,j)$.

Zapiszmy teraz dokładny wzór:

$$P(i,j) = \begin{cases} 0 & , \text{gdy } \text{obs}(i-j) \leq 2 \\ \min_{i<k<j} \max\left(P(i,k), P(k,j), \text{dist}(p_i,p_k), \text{dist}(p_k,p_j)\right) \end{cases}$$

Dla każdego $(i,j)$ wyliczamy go korzystając z poprednio wyliczonych wartości w $O(n)$. Dlatego sumarycznie algorytm zajmie $O(n^3)$.

Na mie pokrótce jak wyliczyć optymalną długość max podścieżki.
Teraz pokaże jak Tatuś wszystkich cały zbiór:

Stwórzmy tablicę pomocniczą Z, w której Z[i][j] będzie mówiło o tym jaki punkt wybraliśmy licząc P(i,j) jako optymalny podział. Wtedy wiemy, że w wynikowym zbiorze będzie odcinek $\{i, Z[i][j]\}$ oraz $\{Z[i][j], j\}$; dodatkowo rekurencyjnie odwołujemy się od Z[i][k] oraz Z[k][j] aby wszystkie. Oczywiście zaczynamy od Z[1][n].