Karol Cidyło

Zadanie 6 z listy 1.

Pseudokod:

```
deleteMaxNumbers(t[], n) {
    count = 0;
    j = n/2;
    i = 0;
    while ((j < n) and (i < n / 2)) {
        if ( 2 * t[i] <= t[j) {
            count++;
             i++;
        }
        j++;
    }

    return count;
}
```

Rozwiązanie działa w czasie O(n) sprawdzimy każdy element z tablicy maksymalnie raz.

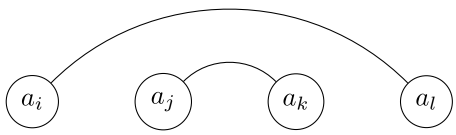
Lemat.
Istnieje optymalne rozwiązanie, które wykreśla pary liczb z ciągu w taki sposób, że pierwszy element z pary jest zawsze w pierwszej połowie ciągu, a drugi w drugiej połowie.
Dowód.
Weźmy optymalne rozwiązanie i nazwijmy je OPT.
Weżmy teraz takie pary liczb z rozwiązania optymalnego, które nie spełniają tego lematu.
Nie możemy mieć sytuacji, w której pierwszy element z pary jest w drugiej połowie, a drugi w pierwszej. Wynika to ze sposobu parowania oraz tego, że mamy niemalejący ciąg liczb dodatnich.
Rozważmy więc sytuacje, w której OPT wybrał pary liczb, których oba elementy należą do tej samej połowy tablicy.
Niech to będzie para (a_i, a_j) dla pierwszej połowy tablicy oraz para (a_k, a_l) dla drugiej połowy tablicy.
Ze sposobu parowania oraz faktu, że mamy niemalejący ciąg liczb wynika:
$2 * a_i <= a_j$ oraz $2 * a_k <= a_l$, wiemy , że skoro a_j jest w pierwszej połowie ciągu, a_k w drugiej to $a_j <= a_k$. Czyli wynika też: $2 * a_i <= a_k$ i $2 * a_j <= a_l$ możemy zatem przekształcić pary wybrane przez OPT do par (a_i , a_k) oraz (a_j, a_l). Po przekształceniu wciąż mamy taką samą liczbę par wykreślonych przez oba algorytmy.
Rozważmy kolejną sytuacje, w której w algorytmie OPT liczba par wybranych w pierwszej połowie jest różna od liczby par wybranych w drugiej połowie.
Weźmy przypadek gdzie w pierwszej połowie jest więcej par, a dokładnie o jedną więcej, niech to będzie para (a_i, a_j), w takim razie w drugiej połowie musi być

jakiś niesparowany element, niech to będzie a_k, $2*a_i <= a_j$ oraz z faktu, że ciąg jest niemalejący $2*a_i <= a_j <= a_k$, więc również $2*a_i <= a_k$, czyli możemy to przekształcić do pary (a_i , a_k). W przypadku gdzie takich par jest więcej powtarzamy powyższe kroki i zawsze będziemy w stanie dopasować element z pierwszej połowy z niesparowanym elementem z prawej połowy. W przypadku odwrotnej sytuacji, gdzie w drugiej połowie jest więcej par stosujemy analogiczne rozwiązanie. W tych przypadkach będziemy w stanie przekształcić pary rozwiązania OPT nie tracąc przy tym żadnego wykreślenia. Algorytm, który wykreśla pary liczb z ciągu, gdzie pierwszy element jest w pierwszej połowie,a drugi w drugiej wykreśla tyle samo par co algorytm OPT.

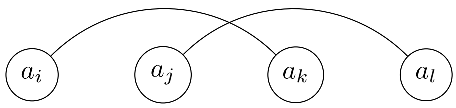
□

Teraz wystarczy pokazać, że pary są dobierane w taki sposób, że pierwszy element z pary(z pierwszej połowy) jest dopasowany z najmniejszym możliwym elementem z drugiej połowy ciągu.
Załóżmy, że i < j < k < l oraz j < n/2 i k >= n/2

Rozważamy takie wykreślone pary:
(a_j, a_k) oraz (a_i, a_l)



W naszym algorytmie najpierw porównywalibyśmy ze sobą a_i oraz a_k zanim doszlibyśmy do porównania a_j oraz a_k Więc też mielibyśmy wykreśloną parę (a_i oraz a_k). Skoro a_j mogliśmy wykreślić z a_k to tak samo możemy wykreślić w parze z a_l



Korzystając z lematu oraz przekształceń takich jak powyżej możemy rozwiązanie algorytmu optymalnego przekształcić do takiego, które zwraca nasz algorytm dając ten sam wynik. Z tego wynika, że algorytm daje optymalne rozwiązanie.