Omówienia zadań z turnieju indywidualnego nr 21

Bartosz Chomiński Marcel Szelwiga

28 września 2024

Chińska gra

Marcel Szelwiga

Zadanie polega na implementacji programu, który pozwoli przewidzieć wynik gry o zasadach podanych w treści zadania i prezentujących się następująco:

- wygrywa figurka o większej mocy,
- każda figurka ma swoją moc i żywioł,
- każda figurka stoi na karcie, która zwiększa lub zmniejsza moc stojącej na niej figurki w zależności od żywiołu,
- niektóre żywioły dominują nad innymi, co przekłada się na podwojoną moc figurki o dominującym żywiole.

W celu rozwiązania zadania należało wyznaczyć moc obu figurek. Moc każdej figurek odpowiada sumie bazowej mocy i wartości bonusu z karty, zależnego od żywiołu danej figurki. Po wyznaczeniu mocy obu figurek, należy podwoić moc jednej z nich w przypadku gdy dochodziło do dominacji żywiołów. Po wyznaczeniu ostatecznej mocy poszczególnych figurek wystarczyło je porównać i wypisać odpowiedź.

Rozważmy teraz kilka szczegółów implementacyjnych. Jeśli wczytamy moc figurki i jej żywioł jako `M` i `T` oraz bonusy karty, na której stoi odpowiednio jako `W`, `O`, `P`, `Z`, to moc figurki możemy wyznaczyć czterema wyrażeniami typu `if`, w których będziemy porównywać żywioł figurki z możliwymi wartościami (`'W'`, `'O'`, `'P'`, `'Z'`) i odpowiednio zwiększali moc `M` figurki. Przykładowo jeden z takich warunków wyglądałby następująco:

```
if T = 'W' then M += W
```

Musimy jeszcze rozważyć dominację żywiołów; załóżmy, że żywioły i moce figurek mamy zapisane odpowiednio w zmiennych `T1`, `T2`, `M1` i `M2`. Dominację żywiołów najprościej wyznaczyć z użyciem ośmiu konstrukcji typu `if`, w których będziemy odpowiednio rozważali wszystkie przypadki dominacji. Jeden z takich warunków wyglądałby następująco:

```
if T1 = 'W' and T2 = 'O' then M1 := M1 * 2
```

Na końcu zostało nam wyznaczenie odpowiedzi, które również powinno składać się z porównań mocy figurek po uwzględnionym bonusie z karty i dominacji żywiołów.

Winda

Bartosz Chomiński

Zmiana zwrotu ruchu windy następuje wtedy, gdy najpierw winda jedzie do góry, a potem w dół lub odwrotnie. Z podanych numerów pięter, na których kolejno zatrzymywała się winda możemy wywnioskować kiedy jechała w górę, a kiedy w dół i zapisać te kierunki w nowej tablicy.

Przykładowo, jeśli winda zatrzymywała się kolejno na piętrach: 8, 3, 4, 5, 7, 6, to w nowej tablicy zapiszemy -1, $+1$, $+1$, $+1$, -1, gdzie $+1$ odpowiada jeździe do góry, a -1 – jeździe w dół. Zwracamy uwagę na to, że nowa tablica jest o jedną komórkę mniejsza od oryginalnej.

W tym celu można skorzystać z pętli typu `for`, w której znajdzie się wyrażenie typu `if` z warunkiem `A[i] < A[i+1]`, od którego będzie zależeć czy komórce nowej tablicy przypiszemy $+1$, czy -1.

Gdy mamy już tablicę z kierunkami ruchu, to wystarczy policzyć ile razy w trakcie przejścia po tej tablicy -1 zmienia się na $+1$ lub odwrotnie. Warto przy tym zwrócić uwagę na ograniczenia dotyczące zmiennej, której wartość wzrasta w pętli.

Alternatywnie: można nie konstruować dodatkowej tablicy i sprawdzać warunek zmiany kierunku bezpośrednio na podanych numerach pięter – winda zmienia zwrot ruchu po zatrzymaniu się na i-tym z kolei piętrze, gdy (`A[i-1] < A[i]` oraz `A[i] > A[i+1]`) LUB (`A[i-1] > A[i]` oraz `A[i] < A[i+1]`), gdzie `A[i]` to numer i-tego z kolei piętra, na którym zatrzymała się winda.

Hieroglify

Marcel Szelwiga

W zadaniu dane jest słowo S złożone z liter `H` i `Y`, na podstawie którego należy wyznaczyć słowo S', które względem słowa S zamienioną ma literę na jednej pozycji i maksymalizuje liczbę wystąpień podciągu `HY`. W przypadku gdy istnieje wiele rozwiązań należy wybrać, to które modyfikuje skrajnie lewą pozycję.

Do zadania można podejść w klasyczny sposób, czyli zastanowić się w jaki sposób zamiana litery na określonej pozycji wpłynie na liczbę wszystkich wystąpień podciągu `HY` i spośród wszystkich zmian wybrać tą, która maksymalizuje różnicę.

Zanim przejdziemy do analizy tego, w jaki sposób zmiana litery wpływa na liczbę wystąpień podciągu `HY` wprowadźmy kilka definicji:

- $\Delta(x)$ – zmiana sumarycznej liczby wystąpień pociągu `HY` po zamianie litery na pozycji x.
- $H(x)$ – liczba wystąpień litery `H` do pozycji x włącznie,
- $Y(x)$ – liczba wystąpień litery `Y` do pozycji x włącznie,
- Y_S – liczba wystąpień litery `Y` w słowie S.

Rozważmy najpierw zamianę literę `H` na `Y` na pozycji x-tej; zauważmy, że liczba podciągów `HY` zmniejszy się o liczbę wystąpień litery `Y` za pozycją x-tą i zwiększy się o liczbę wystąpień litery `H` do pozycji $(x-1)$-wszej, a zatem:

$$\Delta(x) = H(x-1) - (Y_S - Y(x)).$$

Rozważmy teraz zamianę litery `Y` na `H` na pozycji x-tej; zauważmy, że liczba podciągów `HY` zmniejszy się o liczbę wystąpień litery `H` do pozycji x-tej i zwiększy się o liczbę liter `Y` za pozycją x-tą, mamy zatem:

$$\Delta(x) = (Y_S - Y(x)) - H(x).$$

Ostatecznie otrzymujemy zatem zależność:

$$\Delta(x) = \begin{cases} H(x-1) - (Y_S - Y(x)) \text{ dla } S_x = \texttt{H} \\ (Y_S - Y(x)) - H(x) \text{ dla } S_x = \texttt{Y} \end{cases}$$

By uzyskać słowo S' należy wybrać takie minimalne x, dla którego $\Delta(x)$ jest maksymalna i zamienić literę słowa S na pozycji x-tej.

Karol, który został krupierem

Bartosz Chomiński

Zauważmy, że wszystkich możliwych stritów, ale bez uwzględniania kolejności kart w stricie, jest $10 \cdot (4^5 - 4) = 10\,200$. Wobec tego wystarczy odpowiednio szybko wygenerować wszystkie możliwe strity i dla każdego z nich sprawdzić jak dużo kart składających się na tego strita jest już na ręce gracza nr 1. Następnie wybieramy tę opcję, z którą ręka gracza nr 1 ma najwięcej kart wspólnych i wyznaczamy liczbę brakujących kart.

Sugerujemy generować strity jako tablice napisów oznaczających dane karty zgodnie ze schematem opisu podanym w treści zadania – wtedy sprawdzenie równości kart sprowadza się do zwykłego porównania napisów.

Znaki ostrzegawcze

Bartosz Chomiński

Spróbujmy stawiać znaki na słupach, idąc od pierwszego do ostatniego, najwcześniej jak jest to możliwe – czyli jeśli widzimy na słupie wolne miejsce i w odpowiedniej odległości przed nami jest niebezpieczeństwo „niepokryte" jeszcze znakiem, to od razu stawiamy znak ostrzegający o tym niebezpieczeństwie i idziemy dalej.

Pokażemy, że podejście opisane wyżej (znane również jako *zachłanne*) zwróci nam poprawne ustawienie znaków zawsze wtedy, gdy jakiekolwiek poprawne ustawienie istnieje.

Załóżmy, że istnieje poprawne ustawienie znaków – nazwijmy je OPT. Pokażemy w jaki sposób można je zmodyfikować, zachowując cały czas poprawność, by uzyskać ustawienie, które podałby nasz algorytm dla tych samych danych (nazwijmy je ALG).

Niech Z_k oznacza znak dotyczący k-tego z kolei niebezpieczeństwa. Niech S_i oznacza słup, na którym jest znak Z_1 w rozwiązaniu OPT i niech S_j oznacza słup, na którym jest Z_1 w ALG. Jeśli $i = j$, to rozwiązania OPT i ALG przynajmniej w tej części się zgadzają (pozostałą częścią zajmiemy się później); jeśli $i \neq j$, to mamy dwie możliwości: $i < j$ lub $i > j$.

Jeśli $i < j$, to mamy sprzeczność, bo w rozwiązaniu ALG znak Z_1 jest co do zasady najwcześniej jak jest to możliwe (czyli na pierwszym słupie bliższym niż B metrów do pierwszego niebezpieczeństwa), co oznaczałoby, że S_i jest dalej niż B metrów od pierwszego niebezpieczeństwa, czyli rozwiązanie OPT jest niepoprawne. Wobec tego nie musimy się przejmować przypadkiem $i < j$, bo nie ma on prawa wystąpić.

Jeśli natomiast $i > j$, to spróbujemy przestawić znak Z_1 w OPT ze słupa S_i na słup S_j. Jeśli na słupie S_j jest jakieś wolne miejsce, to od razu przestawiamy Z_1 – rozwiązanie po modyfikacji nazwiemy OPT_1 i wiemy o nim, że zgadza się z ALG w części dotyczącej Z_1. Jeśli jednak okaże się, że na S_j nie ma już miejsca, to wystarczy zamienić miejscami Z_1 i dowolny znak z S_j (dowód poniżej) i w ten sposób również uzyskamy lekko zmodyfikowane **poprawne** rozwiązanie OPT_1, które zgadza się z ALG na Z_1.

Oto dowód tego, że dowolny znak z S_j można przenieść na S_i. Niech Z_k oznacza dowolny jeden znak z S_j i niech dotyczy on k-tego niebezpieczeństwa. Dalej oznaczamy, jak w treści zadania, D_k jako pozycję k-tego znaku i S_j jako pozycję j-tego słupa. Wiemy więc, skoro Z_k był na S_j w poprawnym rozwiązaniu OPT, że $A \leq D_k - S_j \leq B$. Wiemy też, że skoro Z_1 wisi w OPT na S_i, to $A \leq D_1 - S_i \leq B$. Z tych czterech nierówności możemy teraz wywnioskować, że

$$A \leq D_1 - S_i \leq D_k - S_i \leq D_k - S_j \leq B,$$

co precyzyjnie oznacza, że znak Z_k może zgodnie z warunkami zadania wisieć na słupie S_i.

Wróćmy więc do głównego wątku: mamy rozwiązanie OPT, rozwiązanie ALG i rozwiązanie OPT_1, w którym znak Z_1 wisi na tym samym słupie co w ALG. Teraz wystarczy podobne rozumowanie przeprowadzić dla znaku Z_2, następnie Z_3 i tak dalej aż małymi kroczkami dojdziemy do rozwiązania OPT_n, które będzie zgadzać się z ALG na wszystkich znakach, czyli te rozwiązania będą identyczne. Tak więc pokazaliśmy, że jeśli istnieje jakiekolwiek ustawienie znaków dla konkretnych danych wejściowych, to nasz algorytm też znajdzie jakieś poprawne ustawienie.

Implementacja tego algorytmu może wyglądać następująco: ustawiamy wskaźnik $i = 0$ na początek tablicy znaków oraz wskaźnik $j = 0$ na początek tablicy słupów. Jeśli znak Z_i może być na słupie S_j, to zapisujemy sobie to przypisanie, zwiększamy i oraz j i kontynuujemy. Jeśli S_j jest przed i-tym niebezpieczeństwem, ale za daleko, to zwiększamy j i kontynuujemy. Jeśli natomiast S_j jest za blisko i-tego niebezpieczeństwa lub za i-tym niebezpieczeństwem, to kończymy wykonanie z wynikiem negatywnym.

Złożoność tego algorytmu jest liniowa – znaki i słupy są już posortowane, więc wystarczy przejść po słupach odpowiednią pętlą i utrzymywać aktualne zapełnienie słupów oraz zbiór niebezpieczeństw zabezpieczonych już znakiem.

Bomby francuskie

Marcel Szelwiga

W zadaniu dana jest plansza o wymiarach $N \times M$ złożona ze znaków `A`, `H` i `.`, opisująca rozmieszczenie fajerwerków dwóch typów na placu. Wiadomo, że fajerwerk typu `A` wysadza wszystkie pola planszy odległe o nie więcej niż R_A w metryce maksimum, a fajerwerk typu `H` wysadza wszystkie pola planszy odległe o nie więcej niż R_H w metryce Manhattan.

Na postawie opisu planszy należy wyznaczyć minimalną sumę R_A i R_H, dla której można dobrać R_A i R_H, w taki sposób, by cała plansza została pokryta eksplozją fajerwerków.

Załóżmy na początek (niezgodnie z prawdą), że w optymalnym rozwiązaniu $R_A = 0$ lub $R_H = 0$. Zauważmy, że w takim przypadku zadanie sprowadziłoby się do wyznaczenia minimalnego R_A, dla którego eksplozja fajerwerków typu `A` pokryje całą planszę lub minimalnego R_H, dla którego eksplozja fajerwerków typu `H` pokryje całą planszę. Biorąc pod uwagę, że zagadnienia te są dość podobne, spróbujmy najpierw rozwiązać jedno z nich – dla fajerwerków typu `H`.

Spróbujmy wyznaczyć dla każdego pola jego odległość do najbliższego fajerwerka typu `H` i wybrać maksymalną z tych odległości jako odpowiedź. Zauważmy, że takie odległości możemy wyznaczyć używając algorytmu BFS z wielu źródeł.

W Algorytmie BFS wierzchołkami startowymi będą wszystkie pola zawierające fajerwerka typu `H`, a dla każdego pola (wierzchołka) jego sąsiadami będą pola sąsiednie (te, z którymi dzieli krawędź).

Zauważmy, że w analogiczny sposób, możemy wyznaczyć dla każdego pola odległości do najbliższego fajerwerka typu `A`. Gdyby więc prawdziwe było nasze początkowe założenie ($R_A = 0$ lub $R_H = 0$), to wystarczyłoby teraz odszukać najbardziej odległe pole od fajerwerka typu `H` i najbardziej odległe pole od fajerwerka typu `A` i wybrać mniejszą spośród maksymalnych odległości jako odpowiedź.

Spróbujmy teraz rozwiązać właściwe zadanie i wykorzystajmy postęp, który zdobyliśmy rozwiązując jego prostszy wariant – dla każdego pola wyznaczyliśmy parę (R_a, R_h) – odpowiednio R_A i R_H niezbędne do pokrycia eksplozją danego pola przez fajerwerki określonych typów. Zauważmy, że gdybyśmy mieli magiczną kulę, która powiedziałaby nam ile wynosi R_A, to potrafilibyśmy prosto wyznaczyć R_H – byłoby to maksimum spośród R_h dla wszystkich par (R_a, R_h), dla których $R_a > R_A$ (czyli maksymalna odległość do najbliższej fajerwerka typu `H` dla pól niepokrytych eksplozją fajerwerka typu `A`). Niestety nie posiadamy czegoś takiego jak magiczna kula, jej odpowiednikiem będzie zatem pętla `for`.

Zauważmy, że po posortowaniu par (R_a, R_h) każda pozycja w tym posortowaniu odpowiadać będzie sytuacji w której prefiks pól jest pokryty przez fajerwerki typu `A`, a sufiks pól nie będzie pokryty przez fajerwerki typu `A`, musi zatem zostać pokryty przez fajerwerki typu `H`, a zatem wiemy, że R_H powinno być równe maksymalnemu R_h na sufiksie.

Otrzymaliśmy w ten sposób prosty sposób na sprawdzenie wszystkich możliwych wartości dla R_A – iterujemy się po posortowanych parach (R_a, R_h) i jako R_A wybieramy aktualne R_a, a jako R_H dobieramy maksymalne R_h na sufiksie.

Otrzymamy tym samym złożoność $\mathcal{O}(N \cdot M \log(N \cdot M))$, wynikającą z konieczności posortowania listy wyznaczonych dwoma BFS-ami par (R_a, R_h).

Taksówkarz

Bartosz Chomiński

Stwórzmy graf skierowany G, w którym wierzchołkami będą początki i końce wszystkich podanych zleceń oraz punkt początkowy i końcowy dnia pracy Karola. Niech każde dwa wierzchołki będą połączone krawędzią z wagą oznaczającą zysk z bezpośredniej podróży z początku do końca tej krawędzi – jeśli krawędź prowadzi od początku do końca jednego zlecenia, to zysk wyliczamy tak, jak gdybyśmy wykonywali to zlecenie; w przeciwnym razie zysk będzie niedodatni i równy liczbie przeciwnej do liczby kilometrów między końcami tej krawędzi.

W tym modelu rozwiązanie zadania sprowadza się do wyznaczenia jak największej odległości między dwoma ustalonymi wierzchołkami w grafie ważonym, w którym krawędzie mogą mieć ujemne lub dodatnie wagi.

W literaturze znane są dwa ważne rozwiązania powyższego zadania: algorytm Bellmana–Forda o złożoności $\mathcal{O}(|V| \cdot |E|)$ oraz algorytm Floyda–Warshalla o złożoności $\mathcal{O}(|V|^3)$.

W kontekście całego zadania oba algorytmy mają identyczną złożoność, ponieważ $|E| \in \mathcal{O}(|V|^2)$, ale możemy zoptymalizować je na różne sposoby.

Podejście związane z algorytmem Floyda–Warshalla możemy przyspieszyć, zauważając że nie potrzebujemy w grafie wierzchołków odpowiadających za początki zleceń – odległość z punktu końcowego zlecenia Z_i do punktu końcowego zlecenia Z_j będziemy wyznaczać rozpatrując dwie możliwości: albo wykonujemy zlecenie Z_j, albo nie. Czasami wykonanie tego zlecenia może nam przynieść większy zysk w kontekście przemieszczenia się z końca Z_i do końca Z_j, a czasami punkt początkowy Z_j może być na tyle daleko, że nie będzie się opłacało wykonywać tego zlecenia. W ten sposób zmniejszamy o połowę liczbę wierzchołków i wówczas mieścimy się w limicie czasu.

Podejście związane z algorytmem Bellmana–Forda możemy usprawnić przez sprytne zmniejszenie liczby krawędzi kosztem lekkiego zwiększenia liczby wierzchołków oraz przez ograniczenie liczby iteracji głównej pętli (odpowiadającej maksymalnej liczbie krawędzi na poprawnie wyliczonych ścieżkach).

Pomysł jest następujący: skoro odległości są liczone w tzw. *metryce Manhattan*, to rozważmy oddzielnie ruch pionowy i ruch poziomy przy przechodzeniu z jednego punktu do drugiego. Będziemy teraz tworzyć nowy zbiór wierzchołków V' i nowy zbiór krawędzi E'. Niech $X = \{x_1, x_2, \ldots, x_m\}$ oznacza zbiór wszystkich odciętych, które pojawiają się w opisach punktów podanych jako dane wejściowe. Niech $U(i) = \{\min\{n \cdot 2^j : n \cdot 2^j \geq i, n \in \mathbb{N}\} : j \in \mathbb{N}\}$ oraz niech $D(i) = \{\max\{n \cdot 2^j : n \cdot 2^j \leq i, n \in \mathbb{N}\} : j \in \mathbb{N}\}$. O zbiorach $U(i)$ oraz $D(i)$ możemy myśleć jak o zbiorach kolejnych zaokrągleń i do coraz wyższych potęg dwójki. Przy takich oznaczeniach niech

$$V' = V \cup \{(x_k, y) : k \in D(i) \cup U(i), k \leq m, (x_i, y) \in V\}$$

oraz niech

$$\begin{aligned}
E' = {} & \{(S_i, E_i) : S_i \text{ oraz } E_i \text{ to początek i koniec } i\text{-tego zlecenia}, i \leq N\} \cup \\
& \cup \{((x_i, y), (x_k, y)) : k \in D(i) \cup U(i), k \leq m, (x_i, y) \in V\} \cup \\
& \cup \{((x_k, y), (x_i, y)) : k \in D(i) \cup U(i), k \leq m, (x_i, y) \in V\} \cup \\
& \cup \{((x_i, y_j), (x_i, y_k)) : (x_i, y_j), (x_i, y_k) \in V', y_j \leq y_k, \neg(\exists y)((x_i, y) \in V' \wedge y_j \leq y \leq y_k)\} \cup \\
& \cup \{((x_i, y_j), (x_i, y_k)) : (x_i, y_j), (x_i, y_k) \in V', y_k \leq y_j, \neg(\exists y)((x_i, y) \in V' \wedge y_k \leq y \leq y_j)\},
\end{aligned}$$

gdzie wagi krawędzi z E' ustalamy zgodnie z zyskiem zlecenia (w pierwszym z pięciu podzbiorów) lub zgodnie z odległością (w czterech pozostałych podzbiorach).

Zauważmy, że każdą krawędź z E możemy utożsamić z ciągiem krawędzi z E', który realizuje tę samą trasę tym samym zyskiem – przykładowo, jeśli w E jest krawędź z (x_5, y) do (x_{19}, y'), to w E' zrealizujemy ją kolejno krawędziami: $(x_5, y) \to (x_{16}, y) \to (x_{19}, y) \to \cdots \to (x_{19}, y')$.

Warto też zauważyć, że liczba krawędzi w zbiorze E' jest rzędu $\mathcal{O}(N \log N)$, gdzie N jest liczbą zleceń, natomiast liczba krawędzi w zbiorze E jest rzędu $\mathcal{O}(N^2)$. Ponadto, jeśli zadbamy o to, by kolejność przechodzenia krawędzi w każdym wywołaniu głównej pętli w algorytmie Bellmana–Forda była odpowiednia (czyli by pionowe krawędzie w obrębie tej samej rzędnej były przechodzone po kolei), to wystarczy nam $\mathcal{O}(N)$ obrotów głównej pętli, bo częściową relaksację jednej krawędzi z E będzie można wykonać trzema obrotami głównej pętli w zmodyfikowanej wersji algorytmu Bellmana–Forda. Uzyskujemy wówczas złożoność $\mathcal{O}(N^2 \log N)$, co przy efektywnym zaimplementowaniu algorytmu powinno pozwolić na odróżnienie od wersji z algorytmem Floyda–Warshalla już od $N \geq 4\,000$, jednak wówczas limit czasu przekraczałby granice zdrowego rozsądku, więc akceptowaliśmy oba opisane rozwiązania.

Wieżowce warszawskie

Marcel Szelwiga

W zadaniu dany jest ciąg N parami różnych liczb parzystych A, a jego celem jest wyznaczenie maksymalnej liczby inwersji, jaką można uzyskać poprzez zamianę jednej wartości w ciągu na dowolną liczbę nieparzystą.

Spróbujmy podejść do tego zadania klasycznie, czyli dla każdej pozycji w ciągu wyznaczyć najlepszą możliwą wartość do zmiany i to o ile zmieni się liczba inwersji całego ciągu po zamianie wartości na określonej pozycji na wartość optymalną na tej pozycji. Zanim jednak przystąpimy do dokładniejszej analizy problemu wprowadźmy dwie definicje:

- $G(x, y, V) = |\{A_i > V : x \leq i \leq y\}|$,
- $S(x, y, V) = |\{A_i < V : x \leq i \leq y\}|$.

Zacznijmy od prostszej rzeczy – wyznaczenia o ile zmniejszy się liczba inwersji po wymazaniu liczby z określonej pozycji. W tym celu rozważmy pozycję x i zauważmy, że po wymazaniu liczby A_x z ciągu liczba inwersji zmniejszy się, o liczbę wszystkich inwersji, utworzonych z liczbą na pozycji x, z liczbami większymi od A_x na lewo od niej i z liczbami mniejszymi od A_x na prawo od niej:

$$G(0, x, A_x) + S(x, N-1, A_x).$$

Do wyznaczenia poszczególnych elementów tej sumy dla każdej pozycji możemy użyć standardowego zamiatania z użyciem drzewa przedziałowego ustaw w punkcie, suma na przedziale – przechodzimy po ciągu od $x = 0$ do $x = N-1$, jako $G(0, x, A_x)$ ustawiamy wynik operacji query(A_x, ∞) w drzewie przedziałowym, a następnie ustawiamy na pozycji A_x wartość 1. Wartości $S(x, N-1, A_x)$ można wyznaczyć w analogiczny sposób, iterując się po ciągu wspak. Zauważmy, że przy okazji wyznaczyliśmy liczbę inwersji w ciągu i to na kilka sposobów:

$$\sum_{x=0}^{N-1} G(0, x, A_x) = \sum_{x=0}^{N-1} S(x, N-1, A_x) = \frac{1}{2} \cdot \left(\sum_{x=0}^{N-1} G(0, x, A_x) + S(x, N-1, A_x) \right)$$

Zastanówmy się teraz w jaki sposób dla każdej pozycji wyznaczyć maksymalną liczbę inwersji jaką możemy dostać poprzez dodanie nowej liczby na pozycji x. Będzie to taka liczba A'_x, która maksymalizuje sumę

$$G(0, x, A'_x) + S(x, N-1, A'_x).$$

Zauważmy, że wartość tej sumy dla najlepszego wyboru A'_x musimy wyznaczyć dla każdej pozycji x. Spróbujmy rozwiązać problem bardziej ogólny i dla każdej pozycji x wyznaczyć wartość tej sumy dla dowolnego A'_x.

Rozważmy pozycję x, zauważmy, że na lewo od x interesują nas liczby większe od A'_x, a na pozycjach na prawo od x interesują nas liczby mniejsze od A'_x. Spróbujmy trochę przekształcić to zdanie. Gdy rozważamy zamianę liczby na pozycji x, to dowolna liczba B na pozycji na lewo od x będzie tworzyła inwersję dla $A_x < B$, natomiast dowolna liczba C na prawo od x będzie tworzyła inwersję dla $A_x > C$; nazwijmy te przypadki odpowiednio *pierwszym* i *drugim*.

Spróbujmy teraz wyznaczać wartości po kolei dla każdego x od 0 do $N-1$, zauważmy, że na początku wszystkie liczby (za wyjątkiem tej na pozycji zerowej będą wpadały w drugi przypadek z powyższych rozważań), a gdy x będzie przesuwał się o jeden w prawo, to jedna

liczba będzie przechodziła do przypadku pierwszego i jedna liczba będzie zabierana z przypadku drugiego.

Zastanówmy się teraz w jaki sposób utrzymywać dla każdego A'_x i pozycji x wartość sumy $G(0, x, A'_x) + S(x, N-1, A'_x)$. Zauważmy, że możemy w tym celu użyć drzewa przedziałowego dodaj na przedziale, maksimum na przedziale. Liczba B z przypadku pierwszego będzie odpowiadała dodaniu na przedziale od B do 0 wartości 1, a liczba C z przypadku drugiego będzie odpowiadała dodaniu na przedziale od C do ∞ wartości 1. Zauważmy, że wartość drzewa przedziałowego w punkcie A'_x będzie odpowiadała wartości sumy $G(0, x, A'_x) + S(x, N-1, A'_x)$.

Można się tutaj zastanowić dlaczego potrzebujemy utrzymywać drzewo dodaj na przedziale, maksimum na przedziale zamiast dodaj na przedziale, wartość w punkcie. Zauważmy jednak, że naszym celem jest szybkie wyznaczenie największej wartości sumy $G(0, x, A'_x) + S(x, N-1, A'_x)$, a więc maksimum w drzewie przedziałowym, a taka operacja nie jest udostępniana przez drugi wariant drzewa.

Gdy potrafimy już wyznaczyć o ile zmniejszy się liczba inwersji po wymazaniu liczby z dowolnej pozycji, jak i o ile może zwiększyć się liczba inwersji po dodaniu dowolnej liczby na dowolną pozycję, to jesteśmy w stanie prosto wyznaczyć odpowiedź na zadanie – znajdując pozycję, dla której różnica jest maksymalna.

Otrzymujemy tym samym złożoność czasową $\mathcal{O}(N \log N)$, wynikającą z faktu wykonywania liniowo wielu operacji na drzewach przedziałowych.