Omówienie zadań AMPPZ 2017

Jury

Instytut Informatyki Uniwersytetu Wrocławskiego

12 listopada 2017

F: Funkcja szyfrująca (autor: Karol Pokorski)

Zadanie

Funkcja $f(x)$ zwraca sumę wszystkich podciągów cyfr liczby x. Znajdź funkcję odwrotną do f.

F: Funkcja szyfrująca (autor: Karol Pokorski)

Zadanie

Funkcja $f(x)$ zwraca sumę wszystkich podciągów cyfr liczby x. Znajdź funkcję odwrotną do f.

- Wzór jawny na wartość $f(x)$ dla n-cyfrowego x:

$$f(x) = 11f\left(\lfloor\frac{x}{10}\rfloor\right) + 2^{n-1} \cdot (x \bmod 10)$$

- Odgadnijmy długość n liczby x.
- Niech $f(x) = r$. Zauważamy, że łatwo odgadnąć drugi składnik powyższej sumy – testujemy brutalnie ile powinno wynosić $x \bmod 10$, żeby $2^{n-1} \cdot (x \bmod 10) = f(x) \bmod 11$.
- Rozwiązanie będzie tylko jedno dlatego, że potęgi dwójki są generatorami grupy $\mathbb{Z}_{11}$.
- Łatwo ustalamy ile wynosi zatem $f\left(\lfloor\frac{x}{10}\rfloor\right)$ co pozwala na iteracyjne odgadnięcie kolejnych cyfr liczby od końca na takiej samej zasadzie jak poprzednio.

G: Głowica drukarki (autor: Karol Pokorski)

Zadanie

Dana jest permutacja. Pokryć ją jak najmniejszą liczbą podciągów monotonicznych o różnicy 1 lub -1.

G: Głowica drukarki (autor: Karol Pokorski)

Zadanie

Dana jest permutacja. Pokryć ją jak najmniejszą liczbą podciągów monotonicznych o różnicy 1 lub -1.

- Jakkolwiek by nie próbować – jedynkę trzeba pokryć tak czy siak. Albo ciągiem rosnącym albo malejącym. W obu przypadkach absolutnie nic nie stoi na przeszkodzie, żeby „przy okazji" pokryć ile tylko jeszcze się da.
- Dla każdego elementu permutacji można zatem wyliczyć jakie wartości pokryjemy ciągiem malejącym i rosnącym. To daje nam przedziały pokrywanych elementów.
- Zadanie teraz sprowadza się do wybrania jak najmniejszej liczby utworzonych przedziałów, aby pokryć wszystkie elementy.

G: Głowica drukarki (autor: Karol Pokorski)

Zadanie

Dana jest permutacja. Pokryć ją jak najmniejszą liczbą podciągów monotonicznych o różnicy 1 lub -1.

- Problem rozwiązujemy zachłannie z użyciem techniki zamiatania. Spośród wszystkich przedziałów pokrywających jedynkę wybieramy ten, który się kończy najpóźniej – niech kończy się na wartości x. Następnie spośród przedziałów rozpoczynających się najpóźniej od $x+1$ wybieramy ponownie ten kończący się najpóźniej. I tak dalej.

E: Edukacja (autor: Karol Pokorski)

Zadanie

Wybrać z losowej połowy prawdziwego słownika zbiór słów, aby każda litera alfabetu wystąpiła dokładnie raz.

E: Edukacja (autor: Karol Pokorski)

Zadanie

Wybrać z losowej połowy prawdziwego słownika zbiór słów, aby każda litera alfabetu wystąpiła dokładnie raz.

- Ustalmy „mądrą” kolejność analizowania słów. Mądra kolejność to na przykład taka, która preferuje wcześniejszą analizę słów zawierających dużo rzadkich liter.
- Utrzymujemy zbiór masek bitowych reprezentujących możliwe do uzyskania podzbiory liter.
- Gdy pamięci lub czasu zaczyna brakować – sprawdzamy dla każdej maski na strukturze czy istnieje jakieś nieprzeanalizowane słowo, dla którego różnica symetryczna jego maski bitowej i wybranej maski ze struktury da pełną maskę 26 jedynek.

I: Intensyfikacja laserowa (autor: Karol Pokorski)

Zadanie

Dana jest prostokątna plansza z niektórymi zablokowanymi polami. Dobrać prawdopodobieństwo p zablokowania pozostałych pól, aby wartość oczekiwana liczby ścieżek w dół/w prawo z lewego górnego do prawego dolnego rogu była równa dokładnie k.

I: Intensyfikacja laserowa (autor: Karol Pokorski)

Zadanie

Dana jest prostokątna plansza z niektórymi zablokowanymi polami. Dobrać prawdopodobieństwo p zablokowania pozostałych pól, aby wartość oczekiwana liczby ścieżek w dół/w prawo z lewego górnego do prawego dolnego rogu była równa dokładnie k.

- W zadaniu trzeba być bardzo ostrożnym przy dokonywaniu obliczeń numerycznych. Wiele sensownych algorytmów nie jest stabilnych numerycznie. Dlatego właśnie arytmetyka dużych liczb już w treści zadania (to było ułatwienie!)
- Dość łatwo zauważamy wyszukiwanie binarne po wyniku (szukanym prawdopodobieństwie p).
- Wypada najpierw sprawdzić czy $p = 1$ daje liczbę ścieżek większą lub równą oczekiwanej liczbie k.
- Można też na początku odsiać przypadki brzegowe z zablokowanym polem startowym i/lub końcowym.

I: Intensyfikacja laserowa (autor: Karol Pokorski)

Zadanie

Dana jest prostokątna plansza z niektórymi zablokowanymi polami. Dobrać prawdopodobieństwo p zablokowania pozostałych pól, aby wartość oczekiwana liczby ścieżek w dół/w prawo z lewego górnego do prawego dolnego rogu była równa dokładnie k.

- Potrzebujemy funkcji, która bierze jako parametr prawdopodobieństwo p i zwraca oczekiwaną wartość liczby ścieżek.
- Zastosujemy programowanie dynamiczne po polach specjalnych (start, koniec i pola zablokowane). Wartością jest oczekiwana liczba dobrych ścieżek do tego pola.

I: Intensyfikacja laserowa (autor: Karol Pokorski)

Zadanie

Dana jest prostokątna plansza z niektórymi zablokowanymi polami. Dobrać prawdopodobieństwo p zablokowania pozostałych pól, aby wartość oczekiwana liczby ścieżek w dół/w prawo z lewego górnego do prawego dolnego rogu była równa dokładnie k.

- Wynik stanu można obliczyć z zasady włączeń i wyłączeń używając prostej kombinatoryki (symbol Newtona).
- Okazuje się, że specyficzna postać wzoru na wynik stanu powoduje, że złożoność obliczania wyniku stanu jest liniowa, a nie wykładnicza.

C: Ciąg identyfikatorów (autor: Karol Pokorski)

Zadanie

Dany jest ciąg cyfr. Wstawić jak najwięcej spacji pomiędzy wybrane cyfry tego ciągu, aby powstał ciąg rosnący.

C: Ciąg identyfikatorów (autor: Karol Pokorski)

Zadanie

Dany jest ciąg cyfr. Wstawić jak najwięcej spacji pomiędzy wybrane cyfry tego ciągu, aby powstał ciąg rosnący.

- Zapomnijmy o potencjalnym problemie zer wiodących uzyskanych liczb (detal implementacyjny).
- Kluczowa obserwacja: niech długość wejścia jest równa n, wówczas długość najdłuższej liczby (poza ostatnią) w optymalnym rozwiązaniu można ograniczyć przez $2\sqrt{n}$ (dowód: indukcyjny).
- Teraz można zastosować programowanie dynamiczne. Stanem jest para: długość rozważonego prefiksu cyfr z wejścia oraz pozycja ostatnio wstawionej spacji. Na mocy obserwacji jest tylko $O(n\sqrt{n})$ stanów, a nie kwadratowo wiele.

C: Ciąg identyfikatorów (autor: Karol Pokorski)

Zadanie

Dany jest ciąg cyfr. Wstawić jak najwięcej spacji pomiędzy wybrane cyfry tego ciągu, aby powstał ciąg rosnący.

- W rozwiązaniu potrzebne jest szybkie weryfikowanie, który z dwóch spójnych podciągów cyfr tej samej długości jest mniejszy leksykograficznie: można do tego na przykład zastosować słownik podsłów bazowych (KMR).

H: Hasła (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Dana jest macierz liter $S[1..n][1..m]$. Wyznaczyć najmniejszą leksykograficznie permutację kolumn, aby macierz czytana wiersz po wierszu była posortowana leksykograficznie.

H: Hasła (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Dana jest macierz liter $S[1..n][1..m]$. Wyznaczyć najmniejszą leksykograficznie permutację kolumn, aby macierz czytana wiersz po wierszu była posortowana leksykograficznie.

- Tworzymy wierzchołki $u_1, u_2, \ldots, u_n$ odpowiadające wierszom i $v_1, v_2, \ldots, v_m$ odpowiadające kolumnom.
- Jeśli $S[i][j] > S[i+1][j]$ to dodajemy krawędź $u_i \longrightarrow v_j$.
- Jeśli $S[i][j] < S[i+1][j]$ to dodajemy krawędź $v_j \longrightarrow u_i$.
- Zauważamy, że szukana permutacja kolumn odpowiada posortowaniu topologicznego grafu i zignorowaniu wierzchołków, które odpowiadają wierszom.

H: Hasła (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Dana jest macierz liter $S[1..n][1..m]$. Wyznaczyć najmniejszą leksykograficznie permutację kolumn, aby macierz czytana wiersz po wierszu była posortowana leksykograficznie.

- Możemy więc zachłannie wyciągać wierzchołek o stopniu wejściowym zero i najmniejszym numerze (jest to istotne tylko w przypadku wierzchołków odpowiadających kolumnom).
- Całkowita złożoność to $O(nm + m \log m)$ ze względu na konieczność użycia kolejki priorytetowej.

A: Alternatywne permutacje (autor: Karol Pokorski)

Zadanie

Podać najmniejszą leksykograficznie kolejność wstawiania jak najmniej kluczy do drzewa BST, aby istniało dokładnie k kolejności generujących to samo drzewo.

A: Alternatywne permutacje (autor: Karol Pokorski)

Zadanie

Podać najmniejszą leksykograficznie kolejność wstawiania jak najmniej kluczy do drzewa BST, aby istniało dokładnie k kolejności generujących to samo drzewo.

- Zastanówmy się najpierw dla ustalonego drzewa, jaka jest liczba permutacji kluczy generujących to drzewo.
- Łatwo z tego uzyskać programowanie dynamiczne:

$$DP[u] = DP[\ell] \cdot DP[r] \cdot \binom{SIZE[\ell + r]}{SIZE[\ell]}$$

- Zauważmy, że we wzorze występuje tylko i wyłącznie mnożenie – wszystkie trzy czynniki muszą być dzielnikami wyniku.

A: Alternatywne permutacje (autor: Karol Pokorski)

Zadanie

Podać najmniejszą leksykograficznie kolejność wstawiania jak najmniej kluczy do drzewa BST, aby istniało dokładnie k kolejności generujących to samo drzewo.

- Dla każdego dzielnika k wyliczamy i spamiętujemy najmniejszy możliwy rozmiar drzewa, dla którego odpowiedź jest równa temu dzielnikowi. Zauważamy, że takich dzielników nie ma wiele (maksymalnie około 4000).
- Dla każdego dzielnika musimy zgadnąć $DP[\ell]$ oraz $DP[r]$. To znów są dzielniki k, więc nie ma ich wiele.
- Po tym zgadnięciu wiemy, że w lewym poddrzewie musi być przynajmniej x wierzchołków, a w prawym y. Musimy teraz wybrać $x' \geqslant x$ oraz $y' \geqslant y$ tak, żeby $\binom{x'+y'}{x'}$ miało ustaloną wartość. Możemy dla każdej możliwej wartości (znów: dzielnika k) stablicować i potem przejrzeć wszystkie możliwości.

A: Alternatywne permutacje (autor: Karol Pokorski)

Zadanie

Podać najmniejszą leksykograficznie kolejność wstawiania jak najmniej kluczy do drzewa BST, aby istniało dokładnie k kolejności generujących to samo drzewo.

- Powyższe rozwiązanie jest być może zbyt wolne: iterujemy się po dzielnikach dzielników dzielników (a potem jeszcze po parach (x', y')).
- Możemy jednak zauważyć, że $\min(x, y) \leqslant 20$.
- Wobec tego możemy spamiętać pośrednie wyniki dla każdego dzielnika k i liczby x. Dla każdej takiej pary zgadujemy dzielnik, a następnie wybieramy $x' \geqslant x$ oraz $y' \geqslant y$.
- Można uniknąć sprawdzania wszystkich możliwych x' oraz y' wykonując odpowiedni preprocessing, jednak jest ich zwykle dość mało i nie zmniejsza to istotnie czasu działania.

D: Dzieci w przedszkolu (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Wybrać $n \leqslant k(k-1)/2$ zbiorów mocy k złożone z liczb $\{1, 2, \ldots, \lfloor 3k^2/2 \rfloor\}$ tak, aby przecięcie dowolnych dwóch z nich miało moc 1.

D: Dzieci w przedszkolu (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Wybrać $n \leqslant k(k-1)/2$ zbiorów mocy k złożone z liczb $\{1, 2, \ldots, \lfloor 3k^2/2 \rfloor\}$ tak, aby przecięcie dowolnych dwóch z nich miało moc 1.

- Weźmy liczbę pierwszą p.
- Wyobraźmy sobie wszystkie funkcje liniowe postaci $f(x) = ax + b \bmod p$, gdzie $a \in \{1, 2, \ldots, p-1\}$ oraz $b \in \{0, 1, \ldots, p-1\}$.
- Dwie takie funkcje przecinają się w dokładnie jednym $x \in \{0, 1, \ldots, p-1\}$ jeśli ich współczynniki a są różne, w przeciwnym przypadku ich przecięcie ich puste.

D: Dzieci w przedszkolu (autor: Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)

Zadanie

Wybrać $n \leqslant k(k-1)/2$ zbiorów mocy k złożone z liczb $\{1, 2, \ldots, \lfloor 3k^2/2 \rfloor\}$ tak, aby przecięcie dowolnych dwóch z nich miało moc 1.

- Dla każdej takiej funkcji konstruujemy zbiór, który koduje współczynnik a oraz wartości w $0, 1, \ldots, p-1$. Liczby $\{1, 2, \ldots, p\}$ są użyte do zakodowania a, liczby $\{p+1, p+2, \ldots, 2p\}$ do zakodowania wartości w 0, i tak dalej.
- Różnych funkcji jest $p(p-1)$, a potrzebnych liczb $p^2 + \lceil k(k-1)/(2p) \rceil$, trzeba więc odpowiednio wybrać p.
- Uwaga na $k = 9$!

L: Leśny kemping (autor: Paweł Gawrychowski)

Zadanie

Mamy cykl na n wierzchołkach z ważonymi krawędziami. Chcemy wybrać $k \leqslant n$ wierzchołków tak, aby zmaksymalizować minimalną odległość między dwoma wybranymi wierzchołkami.

L: Leśny kemping (autor: Paweł Gawrychowski)

Zadanie

Mamy cykl na n wierzchołkach z ważonymi krawędziami. Chcemy wybrać $k \leqslant n$ wierzchołków tak, aby zmaksymalizować minimalną odległość między dwoma wybranymi wierzchołkami.

- Szukamy najmniejszego d, dla którego da się wybrać k wierzchołków. Widać, że możemy wyszukiwać binarnie, trzeba więc tylko umieć szybko sprawdzić czy aktualne d jest wystarczająco małe.
- Na chwilę zapominamy o cyklu i myślimy o ścieżce długości n.
- Dla każdego wierzchołka znajduje pierwszy (zgodnie z ruchem wskazówek zegara) z następnych wierzchołków, który leży w odległości $\geqslant d$. Można to zrobić w sumarycznym czasie $O(n)$ przesuwając dwa palce po ścieżce. Myślimy, że są to możliwe skoki.

L: Leśny kemping (autor: Paweł Gawrychowski)

Zadanie

Mamy cykl na n wierzchołkach z ważonymi krawędziami. Chcemy wybrać $k \leqslant n$ wierzchołków tak, aby zmaksymalizować minimalną odległość między dwoma wybranymi wierzchołkami.

- Po wyznaczeniu skoków liczymy, dla każdego wierzchołka, ile skoków można wykonać zaczynając od niego (i kończąc w wierzchołku, z którego nie da się wykonać żadnego skoku). Można to zrobić w czasie $O(n)$ idąc od prawej do lewej.
- Zgadnijmy pierwszy odwiedzony wierzchołek.
- Jeśli można wykonać przynajmniej $k+1$ skoków z pierwszego wierzchołka to d jest dobre.
- Jeśli można wykonać k skoków to musimy jeszcze sprawdzić czy ostatni z odwiedzonych wierzchołków jest odpowiednio daleko od tego pierwszego.
- Jeśli można wykonać co najwyżej $k-1$ skoków z pierwszego wierzchołka to d nie jest dobre.

K: Karty czipowe (autor: Krzysztof Loryś)

Zadanie

Mamy n wierzchołków na górze i n wierzchołków na dole. i-ty wierzchołek na górze jest połączony krawędzią z $\pi(i)$-tym wierzchołkiem na dole. Wierzchołki na górze i na dole są podzielone na grupy, każda grupa to składa się pewnej liczby kolejnych wierzchołków. Każdą grupę można obrócić lub nie. Naszym celem jest zminimalizowanie liczby warstw, na jakie trzeba podzielić krawędzie. Dwie krawędzie mogą być w tej samej warstwie jeśli nie przecinają się.

K: Karty czipowe (autor: Krzysztof Loryś)

Zadanie

Mamy n wierzchołków na górze i n wierzchołków na dole. i-ty wierzchołek na górze jest połączony krawędzią z $\pi(i)$-tym wierzchołkiem na dole. Wierzchołki na górze i na dole są podzielone na grupy, każda grupa to składa się pewnej liczby kolejnych wierzchołków. Każdą grupę można obrócić lub nie. Naszym celem jest zminimalizowanie liczby warstw, na jakie trzeba podzielić krawędzie. Dwie krawędzie mogą być w tej samej warstwie jeśli nie przecinają się.

- Zapomnijmy o grupach (lub powiedzmy, że każda składa się tylko z jednego wierzchołka). Warstwa odpowiada wtedy podciągowi rosnącemu w π. Minimalna liczba warstw jest więc równa maksymalnemu podciągowi malejącemu.

K: Karty czipowe (autor: Krzysztof Loryś)

- Dla wygody odwróćmy kolejność wierzchołków na dole. Wtedy celem jest dobranie obrotów tak, aby można było wybrać jak najwięcej wzajemnie nieprzecinających się krawędzi.
- Stosujemy programowanie dynamiczne. Dla każdego prefiksu długości x wierzchołków na górze i prefiksu długości y wierzchołków na dole liczymy największą liczbę wzajemnie nieprzecinających się krawędzi, których końce należą do tych prefiksów.
- Dodatkowo musimy pamiętać czy blok do którego należy x-ty wierzchołek na górze został obrócony czy też nie (bo inaczej nie wiemy jaki jest to dokładnie element), podobnie dla wierzchołków na dole. Dla wcześniejszych bloków nie podjęliśmy jeszcze żadnej decyzji.

K: Karty czipowe (autor: Krzysztof Loryś)

- Teraz widać, że mamy tylko trzy możliwości: weźmiemy krawędź łączący x-ty wierzchołek na górze z y-tym wierzchołkiem na dole, lub nie użyjemy x-tego wierzchołka na górze, lub nie użyjemy y-tego wierzchołka na dole.
- Dla każdego $x, y \in \{1, 2, \ldots, n\}$ oraz $f, f' \in \{\mathit{false}, \mathit{true}\}$ możemy więc wyliczyć wynik w czasie $O(1)$ korzystając z wcześniej wyznaczonych wyników.
- Limit pamięci był dość mały, trzeba więc było wyznaczać wyniki wiersz po wierszu (to znaczy dla kolejnych wartości $x = 1, 2, \ldots, n$).

J: Jesienne porządki (autor: Paweł Gawrychowski)

Zadanie

Danych jest n liczb. Na ile sposób można wybrać k z nich tak, aby ich suma była podzielna przez r?

J: Jesienne porządki (autor: Paweł Gawrychowski)

Zadanie

Danych jest n liczb. Na ile sposób można wybrać k z nich tak, aby ich suma była podzielna przez r?

- Widać, że istotne jest tylko to ile mamy liczb dających resztę a modulo r, dla $a = 0, 1, \ldots, r-1$, oznaczmy to przez $CNT[a+1]$.
- Dla kolejnych $a = 0, 1, \ldots, r-1$ wyznaczamy $DP[a][i][j]$: ilość sposobów, na które można wybrać i liczb spośród wszystkich liczb dających reszty $0, 1, \ldots, a$ modulo r tak, aby ich suma dawała resztę j modulo r.
- Dla każdego a, i, j musimy następnie zgadnąć ile liczb dających resztę $a+1$ modulo r wybieramy, oznaczmy tę wartość przez $i' \leqslant CNT[a+1]$. Następnie należy zwiększyć $DP[a+1][i+i'][(j+i'(a+1)) \bmod r]$ o $DP[a][i][j] \cdot \binom{CNT[a+1]}{i'}$.

J: Jesienne porządki (autor: Paweł Gawrychowski)

Zadanie

Danych jest n liczb. Na ile sposób można wybrać k z nich tak, aby ich suma była podzielna przez r?

- To daje $O(k^2 r^2)$. Aby uzyskać szybsze rozwiązanie zauważamy, że tak naprawdę wielokrotnie wykonujemy splot dwóch tablic $A[0..k][0..r]$ i $B[0..k][0..r]$.
- A to już policzona tablica $DP[a]$, a $B[i'][j(a+1) \bmod r] = \binom{CNT[a+1]}{i'}$.
- Wynikiem splotu jest tablica $C[0..k][0..r]$, która uzyskujemy zwiększając $C[i+i'][(j+j') \bmod r]$ o $B[i][j] \cdot C[i'][j']$.
- ...równie dobrze możemy zwiększać $C[i+i'][j+j']$ o $B[i][j] \cdot C[i'][j']$, a potem zawinąć uzyskaną tablicę.

J: Jesienne porządki (autor: Paweł Gawrychowski)

Zadanie

Danych jest n liczb. Na ile sposób można wybrać k z nich tak, aby ich suma była podzielna przez r?

- Zauważamy, że ten splot to tak naprawdę mnożenie wielomianów $\sum_{i,j} B[i][j]x^i y^j$ oraz $\sum_{i,j} C[i][j]x^i y^j$.
- Są to wielomiany dwóch zmiennych, ale mimo to można je szybko mnożyć używając FFT: trzeba tylko odpowiednio zamienić je na wielomiany jednej zmiennej (większego stopnia). Daje to złożoność $O(kr^2 \log(kr))$.
- Można też było skorzystać ze specyficznych własności tablicy C i mnożyć wielomiany stopnia $O(k)$ zamiast $O(kr)$.

B: Bajki (autor: Krzysztof Loryś)

Zadanie

Dla danej tablicy $A[1..n]$, ile jest w niej bardzo ciekawych przedziałów? Przedział jest bardzo ciekawy jeśli wszystkie zawarte w nim przedziały są ciekawe. Przedział jest ciekawy jeśli występuje w nim unikalny element.

B: Bajki (autor: Krzysztof Loryś)

Zadanie

Dla danej tablicy $A[1..n]$, ile jest w niej bardzo ciekawych przedziałów? Przedział jest bardzo ciekawy jeśli wszystkie zawarte w nim przedziały są ciekawe. Przedział jest ciekawy jeśli występuje w nim unikalny element.

- Wystarczy wygenerować wszystkie minimalne (w sensie zawierania) przedziały, które nie są ciekawe.
- Dla każdego możliwego końca j wyznaczamy największe $i \leqslant j$, dla którego $A[i..j]$ nie jest ciekawe.
- Rozważmy dwa ostatnie wystąpienia tego samego elementu w $A[1..j]$, oznaczmy ich pozycje przez k, k' (jeśli jest tylko jedno wystąpienie, myślmy że $k = 0$). Wtedy wszystkie przedziały $A[k+1..i], A[k+2..i], \ldots, A[k'..i]$ są ciekawe.

B: Bajki (autor: Krzysztof Loryś)

Zadanie

Dla danej tablicy $A[1..n]$, ile jest w niej bardzo ciekawych przedziałów? Przedział jest bardzo ciekawy jeśli wszystkie zawarte w nim przedziały są ciekawe. Przedział jest ciekawy jeśli występuje w nim unikalny element.

- Musimy więc tylko dla każdego unikalnego elementu dorzucić do struktury przedział (k, k'), a następnie wyciągnąć największą liczbę $i \leqslant j$, która nie należy do żadnego takiego przedziału.
- Utrzymujemy tablicę $B[1..n]$ początkowo wypełnioną zerami. Dla każdego przedziału (k, k') zwiększamy wartości $B[k+1], B[k+2], \ldots, B[k']$ o jeden. Następnie wyciągamy ostatnią pozycję $i \leqslant j$, dla której $B[i] = 0$.
- Tablicę przechowujemy w drzewie binarnym, czas każdej operacji to wtedy $O(\log n)$.