Ezoteryczne Kartki

Wyszukiwania

Jakub Bachurski

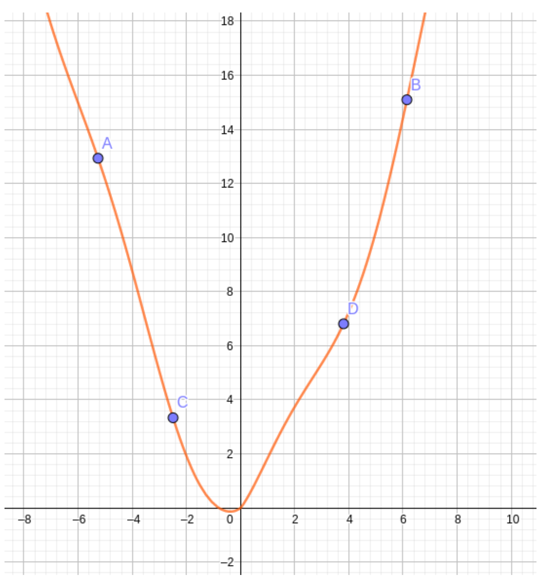
wersja 1.1.2

1 Wyszukiwanie ternarne – ternary search

W wyszukiwaniu binarnym korzystaliśmy z motoniczności – pewna funkcja przez cały czas spełniała pewną relację $\leqslant$. Jest inna ciekawa gałąź funkcji – funkcje bitoniczne. W skrócie są to funkcje, które najpierw rosną, a potem maleją (lub na odwrót). Wyszukiwanie ternarne posłuży nam do znalezienia ekstremum (minimum lub maksimum) takiej funkcji.

1.1 Algorytm

Powiedzmy, że na pewnym zakresie $[a, b]$ poszukujemy minimum pewnej funkcji bitonicznej. Wyszukiwanie ternarne zmniejszy nam ten zakres, ograniczając poszukiwania. Dokona tego poprzez wybranie takich punktów (c, d), że dzielą one zakres na trzy części. Licząc wartości funkcji w punktach a, b, c, d będziemy w stanie zawęzić zakres.



Spójrzmy na przykładowy rysunek: ponieważ $f(a) > f(c)$ i $f(c) < f(d)$, ekstremum musi być gdzieś pomiędzy, bo monotoniczność zmieniła się. Zatem zawężamy zakres do $[a, d]$. Analogicznie, gdyby $f(c) > f(d)$ zawęzilibyśmy zakres do $[c, b]$. Jeżeli szukalibyśmy maksimum, warunki odwróciłyby się (moglibyśmy wtedy rozpatrywać funkcję $f_1(x) = -f(x)$).

Najbardziej opłaca nam się dzielić zakres na trzy równe części, aby zawsze zmniejszać go o $\frac{1}{3}$. Wtedy złożoność wyniesie $O(log_{\frac{3}{2}} \frac{b_0 - a_0}{\epsilon})$, gdzie $[a_0, b_0]$ to początkowy zakres, a ϵ to wielkość przedziału, którą traktujemy jako dostatecznie małą.

1.2 Funkcje wypukłe

Fajnym podzbiorem funkcji bitonicznych są funkcje wypukłe. Też się je w miarę często spotyka, a mają przydatne własności. W skrócie, są to funkcje które rosną ciągle tak samo lub coraz szybciej ($f''(x) \geqslant 0$). Przydadzą nam się te dwie własności:

- Funkcja $h(x) = f(x) + g(x)$ dla funkcji wypukłych f, g jest wypukła.
- Funkcja $h(x) = \max(f(x), g(x))$ dla funkcji wypukłych f, g jest wypukła.

Przykładem funkcji wypukłej jest $f(x) = x^2$.

Funkcja wklęsła to taka, że jej odbicie względem osi OX jest funkcją wypukłą.

1.3 Na liczbach całkowitych

Jeżeli funkcja jest tylko z liczb całkowitych, można użyć zmodyfikowanego wyszukiwania binarnego. Uruchamiamy go po prostu na funkcji

$$g(x) = [f(x) \leqslant f(x+1)]$$

i szukamy pierwszego parametru, dla którego jest to wyrażenie prawdziwe.

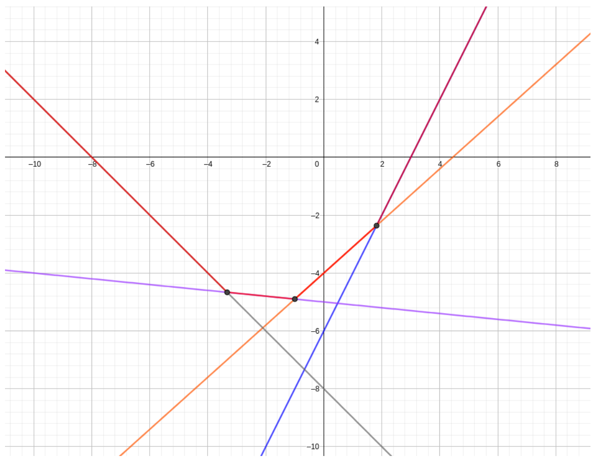
1.4 Zadanka

1.4.1 [PA 2013] Świetliki

Oba zadania zapożyczone od Marka Sommera. Więcej zadań o wyszukiwaniu ternarnym: `http://mareksom.w.staszic.waw.pl/kolko/2017-2018/kartki/22.11.2017-teoria.pdf`

Treść Na płaszczyźnie lata n świetlików – i-ty z nich znajduje się na początku na współrzędnych (x_i, y_i) i ma wektor prędkości (a_i, b_i). Po czasie t znajdzie się w punkcie $(x_i + a_i t, y_i + b_i t)$. Niech d oznacza najmniejszy bok kwadratu o bokach równoległych do osi układu współrzędnych, którym można objąć wszystkie świetliki. Chcemy znaleźć taki czas t, że wartość d jest minimalna.

Rozwiązanie Rozbijmy na razie zadanie na jeden wymiar – przyjmijmy $y_i = b_i = 0$. Świetliki są na osi, a kwadrat spłaszczył się do odcinka. d wyraża się przez $x_{max} - x_{min}$. Przyjrzyjmy się, jak wygląda wykres x_{max} względem t: dla małych t dominują pozycje świetlików które zaczęły z dużym x_i, ale możliwe że mają $a_i < 0$. Z czasem będą prześcigane przez świetliki z większym a_i. Na rysunku wykres czerwony reprezentuje x_{max} – jest on maksimum po wykresach świetlika szarego, fioletowego, pomarańczowego i niebieskiego.



Formalniej, $x_{max} = \max_i x_i + a_i t$, czyli maksimum po funkcjach liniowych – w szczególności funkcje liniowe są wypukłe, zatem funkcja x_{max} jest wypukła.

Przemyślenia są analogiczne dla x_{min}, z tym że wykres jest wklęsły – w takim razie $-x_{min}$ jest wypukłe. Na podstawie tego wywnioskowaliśmy, że $d(t) = x_{max} + (-x_{min})$ jest wypukłe, bo jest sumą funkcji wypukłych.

Na koniec pozostaje rozpatrzeć kwestię drugiej współrzędnej. Rozwiązujemy analogicznie. Powiedzmy że mamy dwie funkcje: $d_x(t)$ oraz $d_y(t)$, czyli d wyznaczone względem x i y. Jasnym jest, że $d(t) = \max(d_x(t), d_y(t))$. A maksimum funkcji wypukłych jest wypukłe, więc $d(t)$ jest wypukłe i możemy po nim wyszukiwać ternarnie. Otrzymujemy w ten sposób złożoność $O(n \log T)$, gdzie T to wyznaczony w arbitralny sposób największy możliwy wynik.

2 Równoległe wyszukiwanie binarne

Równoległe wyszukiwanie binarne jest wariacją wyszukiwania binarnego po wyniku. Będzie on dotyczył zadań w których oczekujemy na pewno wydarzenie *(trigger)*, które w pewnym momencie zajdzie, i możemy dla pewnego momentu czasu, czy wydarzenie już zaszło. Problem pojawia się, gdy chcemy przechwycić wiele podobnych zdarzeń – z pomocą przychodzi równoległy binsercz. A teraz dosyć abstrakcji, przechodzimy do zadanek.

2.1 Zadanka

2.1.1 Moment zespójnienia

Treść Mamy sobie graf o n wierzchołkach i listę m krawędzi E. Każda krawędź charakteryzowana jest przez (u_i, v_i, t_i) – wierzchołki, które łączy (nieskierowanie), oraz czas, w którym staje się aktywna. Mamy też q zapytań, które pytają nas, w którym najwcześniejszym momencie τ_j pewne dwa wierzchołki a_j, b_j są w tej samej spójnej składowej (istnieje pomiędzy nimi ścieżka).

Rozwiązanie Gdyby zapytanie było tylko jedno wystarczyłoby zrobić Find & Union (Disjoint-Seta) na wierzchołkach i odpowiednio je łączyć, posortowawszy krawędzie po czasie aktywacji. Ważnym jest, że krawędzie nie znikają, a wynik τ możemy wyszukiwać binarnie. Niestety, wyszukując binarnie, i tak musimy w $O(m \cdot \alpha(n))$ zasymulować cały proces, więc to podejście nie ma sensu w przypadku jednego zapytania, i wydawałoby się, że również przy wielu. A może jednak...?

Dla każdego z q zapytań wyznaczmy zakres $[p_j, q_j)$ reprezentujący, w jakim przedziale zawiera się τ_j. Na początku tym zakresem jest $[0, T+1]$, gdzie T to maksymalny czas aktywacji (powiedzmy, że jeżeli wyjdzie nam wynik $\tau_j = T+1$, to znaczy że wierzchołki zawsze są w różnych spójnych). Podobnie jak w binserczu strzelimy w wynik $r_j = \lfloor \frac{p_j+q_j}{2} \rfloor$.

Jednak tutaj jest twist: każde z zapytań zostawi *trigger*, który powie nam: „jeżeli jesteś teraz w chwili r_j, to sprawdź czy wierzchołki u_j i v_j są połączone, i powiedz mi". Zatem normalnie zasymulujemy w $O(m \cdot \alpha(n))$ proces dodawania krawędzi, a każde zapytanie ma teraz odpowiedź na nurtujące je pytanie „czy mój wynik τ_j jest co najmniej taki jak r_j?". Za jego pomocą odpowiednio ograniczy swój zakres o połowę, tak samo jak w wyszukiwaniu binarnym. Sprawdzenia zadziałają w $O(q \cdot \alpha(n))$.

Otrzymaliśmy rozwiązanie w $O(\log T \cdot (m + q) \cdot \alpha(n))$.

2.1.2 [Finał XVIII OI] Meteory

Treść Na wyjątkowo niebezpiecznej planecie codziennością są deszcze meteorów. Nieroztropnie wybudowano tam n domków, ponumerowanych od 1 do n, każdy o wytrzymałości na meteory r_i[1]. W związku z prognozowanymi m deszczami meteorów obejmującymi domki o numerach od x_j do y_j o sile s_j i przebiegającymi w momencie t_j, właściciele domków chcą wiedzieć, w którym momencie czasu τ_i będą zmuszeni przeprowadzić ewakuację (z powodu zniszczenia ich domku). Domek jest zniszczony, gdy suma s_j opadów, które go obejmowały, przekracza r_i.

Rozwiązanie Teraz pójdzie o wiele prościej. Deszcze meteorów możemy symulować strukturą danych dodającą na przedziale i pozwalającą na zapytania na

[1] oryginalna treść sugeruje że są to stacje badawcze zbierające meteory, ale bądźmy poważni – „badacze" nie są świadomi zagrożenia, jakie stanowią regularne deszcze meteorów.

punkcie (np. drzewo przedziałowe przedział-punkt albo drzewo Fenwicka) w złożoności $O(\log n)$. Triggery są postaci „czy w punkcie i spadło już r_i meteorów?”, i wkładamy je w pewien czas z_i, będący kandydatem na τ_i. Wystarczy sprawdzić wartość w punkcie naszej struktury. Sumaryczna złożoność rzędu $O(m \log^2 n)$.

3 Wartości i wagi

Na koniec prosty trik z wyszukiwaniem binarnym po wyniku. Wyobraźmy sobie, że sprowadziliśmy zadanie do zbierania przedmiotów o wartościach v_i i wagach w_i na pewnych zasadach, i chcemy teraz zmaksymalizować wynik α:

$$\alpha = \frac{\sum v_i}{\sum w_i}$$

Można się tutaj łatwo złapać w pułapkę podejścia zachłannego, ale zamiast tam utknąć warto zastanowić się nad wyszukiwaniem binarnym po wyniku. Zastanówmy się, czy umiemy otrzymać wynik równy co najmniej α i przekształćmy:

$$\begin{aligned} \frac{\sum v_i}{\sum w_i} &\geqslant \alpha \\ \sum v_i &\geqslant \alpha \sum w_i \\ \sum v_i - \alpha \sum w_i &\geqslant 0 \end{aligned}$$

Przekształćmy teraz nasze wartości i pozbądźmy się wag:

$$v_i' = v_i - \alpha w_i$$

Zauważmy, że teraz $\sum v_i' = \sum v_i - \alpha \sum w_i$. Uprościliśmy zatem zadanie do wybrania przedmiotów na takich samych zasadach, ale teraz mają tylko wartości i chcemy znaleźć dowolny podzbiór sumujący się do nieujemnej liczby.

3.1 Zadanka

Dwa proste zadania wykorzystujące ten koncept.

3.1.1 [KI] Diamenty

Treść W tym zadaniu mamy dane n diamentów o wartościach v_i i wagach w_i. Naszym zadaniem jest wybrać podzbiór rozmiaru k maksymalizujący

$$\frac{\sum v_i}{\sum w_i}$$

Rozwiązanie Aplikujemy trik, rozważamy teraz dla pewnego α istnienie podzbioru wielkości k o nieujemnej sumie spośród liczb $v'_i = v_i - \alpha w_i$. Jest to trywialne – wystarczy je posortować i sprawdzić czy k największych to spełnia. Można też użyć `nth_element`, aby nie narzucać czynnika $O(\log n)$ w złożoności. Wtedy sumaryczna złożoność jest rzędu $O(nT)$, gdzie T to liczba iteracji[2] wybrana jako wystarczająca, by znaleźć optymalny wynik.

3.1.2 [Warsztaty finałowe 2019] Podróżujący kupiec

Treść Mamy graf skierowany, w którym krawędzie mają przydzielone wartości v_i oraz czasy przejścia t_i. Naszym zadaniem jest znaleźć taki cykl, w którym krawędzie, które do niego należą, maksymalizują $\frac{\sum v_i}{\sum t_i}$.

Rozwiązanie Tworzymy nowe wagi $v'_i = v_i - \alpha t_i$ i ponownie rozpatrujemy naszą nierówność:

$$\sum v_i - \alpha \sum t_i \geqslant 0 \iff \sum v'_i \geqslant 0$$

Zastanówmy się nad jej znaczeniem: chcemy znaleźć taki cykl, że suma wag krawędzi na nim jest nieujemna. Rozważmy zamiast tego problem dualny w grafie, w którym wagi krawędzi są zanegowane ($v''_i = -v'_i$). Otrzymujemy

$$\sum v'_i \geqslant 0 \iff \sum v''_i < 0$$

A problem znalezienia cyklu, w którym suma wag krawędzi jest ujemna, umiemy łatwo rozwiązać algorytmem Bellmana-Forda w $O(|V||E|)$. Właśnie stąd takie przekształcenie – ujemny cykl umiemy rozwiązać. Kojarzy nam się z rzeczami możliwe najkrótszymi, a algorytmy na najkrótsze ścieżki znamy. Sumaryczna złożoność $O(|V||E| \cdot T)$, gdzie T to liczba iteracji.

---

[2] W wyszukiwaniu binarnym każda iteracja ogranicza zakres na którym może znajdować się wynik o połowę. Korzystając z triku warto uważać na liczbę iteracji – testowane α właściwie musimy przechowywać jako liczbę zmiennoprzecinkową, a możliwe, że będzie się od nas wymagać znalezienia dokładnego ułamka.