Двоично търсене по отговора

Иван Тончев, ivan.tonchev96@gmail.com

Национална лагер-школа по информатика, Габрово, 2026 г.

"Класическо" двоично търсене

Задача

- Дадено:

- Дадено:
  - **сортирана** редица $A_1 \leq \ldots \leq A_n$

- Дадено:
  - **сортирана** редица $A_1 \leq \ldots \leq A_n$
  - стойност x

- Дадено:
  - **сортирана** редица $A_1 \leq \ldots \leq A_n$
  - стойност x
- Търси се:

- Дадено:
  - **сортирана** редица $A_1 \leq \ldots \leq A_n$
  - стойност x
- Търси се:
  - среща ли се x в A?

- Дадено:
  - **сортирана** редица $A_1 \leq \ldots \leq A_n$
  - стойност x
- Търси се:
  - среща ли се x в A?
  - ако да - някой/най-малък/най-голям индекс на срещане

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$
- Ако $A[mid] == x$, то сме намерили x и сме готови

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$
- Ако $A[mid] == x$, то сме намерили x и сме готови
- Ако $A[mid] < x$, то x със сигурност не се среща и в частта $[low, \ldots, mid - 1]$ и можем да продължим да го търсим в $[mid, \ldots, high]$

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$
- Ако $A[mid] == x$, то сме намерили x и сме готови
- Ако $A[mid] < x$, то x със сигурност не се среща и в частта $[low, \ldots, mid - 1]$ и можем да продължим да го търсим в $[mid, \ldots, high]$
- Ако $A[mid] > x$, то аналогично можем да продължим да го търсим в $[low, \ldots, mid]$

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$
- Ако $A[mid] == x$, то сме намерили x и сме готови
- Ако $A[mid] < x$, то x със сигурност не се среща и в частта $[low, \ldots, mid - 1]$ и можем да продължим да го търсим в $[mid, \ldots, high]$
- Ако $A[mid] > x$, то аналогично можем да продължим да го търсим в $[low, \ldots, mid]$
- Ако low и $high$ се "срещнат" без да сме намерили x, то не се среща в A

"Класическо" двоично търсене

Решение

- Поддържаме подинтервал $[low, \ldots, high]$ на A, в който потенциално можем да намерим x
  - В началото $low = 1, high = n$
- На всяка стъпка проверяваме стойността в средата на интервала $[low, \ldots, high]$
  - $mid := (low + high)/2$
- Ако $A[mid] == x$, то сме намерили x и сме готови
- Ако $A[mid] < x$, то x със сигурност не се среща и в частта $[low, \ldots, mid - 1]$ и можем да продължим да го търсим в $[mid, \ldots, high]$
- Ако $A[mid] > x$, то аналогично можем да продължим да го търсим в $[low, \ldots, mid]$
- Ако low и $high$ се "срещнат" без да сме намерили x, то не се среща в A
- Сложност - $O(log(n))$. Понеже на всяка стъпка дължината на интервала се намаля горе-долу двойно.

Двоично търсене "по отговора"

Примерна задача

Дадени са ни N ($N \leq 10^5$) числа със стойности между -10^9 и 10^9 и число K ($2 \leq K \leq N$). Трябва така да изберем K от числата, така че минималната разлика между две избрани числа да бъде възможно най-голяма.

https://www.geeksforgeeks.org/dsa/place-k-elements-such-that-minimum-distance-is-maximized/

Двоично търсене ”по отговора”

Примерна задача

- Удобно е да имаме числата в сортиран вид. Нека след сортиране, те са $A_1, \ldots, A_n$.

- Удобно е да имаме числата в сортиран вид. Нека след сортиране, те са $A_1, \ldots, A_n$.
- Доста трудно е "директно" да решим задачата. Ако се опитаме да преглеждаме елементите на A един по един и на база техните стойности да увеличаваме/намаляваме потенциалния отговор, то в много случаи може да ни се наложи да ходим напред-назад из редицата и да правим повторни проверки с новата стойност.

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.
  - Монотонна значи сортирана - строго/нестрого намаляваща/растяща.

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.
  - Монотонна значи сортирана - строго/нестрого намаляваща/растяща.
- Те изглеждат много просто - префикс от стойности *false*, последван от суфикс от стойности *true* (при растящите) - или наобратно (при намаляващите)

1	$\ldots$	i	$i+1$	$\ldots$	n
F	$\ldots$	F	T	$\ldots$	T

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.
  - Монотонна значи сортирана - строго/нестрого намаляваща/растяща.
- Те изглеждат много просто - префикс от стойности *false*, последван от суфикс от стойности *true* (при растящите) - или наобратно (при намаляващите)

1	...	i	$i+1$	...	n
F	...	F	T	...	T

  - От тук нататък ще говорим за растящи редици, но нещата при намаляващите е абсолютно аналогично – само трябва да се сменят знаците/посоките

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.
  - Монотонна значи сортирана - строго/нестрого намаляваща/растяща.
- Те изглеждат много просто - префикс от стойности *false*, последван от суфикс от стойности *true* (при растящите) - или наобратно (при намаляващите)

1	...	i	$i+1$	...	n
F	...	F	T	...	T

  - От тук нататък ще говорим за растящи редици, но нещата при намаляващите е абсолютно аналогично – само трябва да се сменят знаците/посоките
- Не е нужно да имаме цялата редица изчислена поначало - достатъчно ще бъде да можем да изчисляваме нейните членове "на момента" , когато ни потрябват.

Двоично търсене "по отговора"

Монотонни редици

- Основа при двоичното търсене "по отговора" са "монотонните" редици от *bool* стойности.
  - Монотонна значи сортирана - строго/нестрого намаляваща/растяща.
- Те изглеждат много просто - префикс от стойности *false*, последван от суфикс от стойности *true* (при растящите) - или наобратно (при намаляващите)

1	...	i	$i+1$	...	n
F	...	F	T	...	T

  - От тук нататък ще говорим за растящи редици, но нещата при намаляващите е абсолютно аналогично – само трябва да се сменят знаците/посоките
- Не е нужно да имаме цялата редица изчислена поначало - достатъчно ще бъде да можем да изчисляваме нейните членове "на момента" , когато ни потрябват.
  - Най-често с функция от вида `bool check(int x)`

Двоично търсене ”по отговора”

Монотонни редици

- Ще се опитваме да свеждаме нашите задачи до това да измислим подходяща такава редица и да намерим ”границата” между *true* и *false*

Двоично търсене ”по отговора”

Монотонни редици

- Ще се опитваме да свеждаме нашите задачи до това да измислим подходяща такава редица и да намерим ”границата” между *true* и *false*
  - В случая на предната задача ще дефинираме редицата по следния начин: ”$f(x) == true$ когато можем да изберем K числа, така че разликата между всеки две от тях да е поне x”

Двоично търсене "по отговора"

Монотонни редици

- Ще се опитваме да свеждаме нашите задачи до това да измислим подходяща такава редица и да намерим "границата" между *true* и *false*
  - В случая на предната задача ще дефинираме редицата по следния начин: "$f(x) == true$ когато можем да изберем K числа, така че разликата между всеки две от тях да е поне x"
  - Така дефинираната редица f е монотонна (намаляваща) - ако например $f(5) == true$, то и за стойностите по-малки от 5 същото е изпълнено

Двоично търсене "по отговора"

Монотонни редици

- Ще се опитваме да свеждаме нашите задачи до това да измислим подходяща такава редица и да намерим "границата" между *true* и *false*
  - В случая на предната задача ще дефинираме редицата по следния начин: "$f(x) == true$ когато можем да изберем K числа, така че разликата между всеки две от тях да е поне x"
  - Така дефинираната редица f е монотонна (намаляваща) - ако например $f(5) == true$, то и за стойностите по-малки от 5 същото е изпълнено
  - Отговорът на задачата е всъщност най-голямото x, за което $f(x) == true$ - тоест достатъчно ни е да намерим границата между *true* и *false* стойностите в f

Двоично търсене "по отговора"

Монотонни редици

- Ще се опитваме да свеждаме нашите задачи до това да измислим подходяща такава редица и да намерим "границата" между *true* и *false*
  - В случая на предната задача ще дефинираме редицата по следния начин: "$f(x) == true$ когато можем да изберем K числа, така че разликата между всеки две от тях да е поне x"
  - Така дефинираната редица f е монотонна (намаляваща) - ако например $f(5) == true$, то и за стойностите по-малки от 5 същото е изпълнено
  - Отговорът на задачата е всъщност най-голямото x, за което $f(x) == true$ - тоест достатъчно ни е да намерим границата между *true* и *false* стойностите в f
  - Сравнително лесно можем да пресмятаме $f(x)$ за дадено x - чрез $O(N)$ грйиди в редицата A (след като първоначално сме я сортирали)

Двоично търсене ”по отговора”

Намиране на границата

Двоично търсене ”по отговора”

Намиране на границата

- Подобно на ”класическото” двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото *low* и *high* са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$
- Взимаме средата $mid := (low + high)/2$

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$
- Взимаме средата $mid := (low + high)/2$
- Ако $f(mid) == true$, то за следващата стъпка $high := mid$

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$
- Взимаме средата $mid := (low + high)/2$
- Ако $f(mid) == true$, то за следващата стъпка $high := mid$
- Ако $f(mid) == false$, то за следващата стъпка $low := mid$

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$
- Взимаме средата $mid := (low + high)/2$
- Ако $f(mid) == true$, то за следващата стъпка $high := mid$
- Ако $f(mid) == false$, то за следващата стъпка $low := mid$
- Ако low и $high$ се "срещнат", тоест $low + 1 == high$, то сме готови

Двоично търсене "по отговора"

Намиране на границата

- Подобно на "класическото" двоично търсене ще поддържаме подинтервал $[low, \ldots, high]$, в който сме сигурни че е границата
  - В началото low и $high$ са най-малката възможна и най-голямата възможна стойност (в зависимост от ограниченията на задачата)
- На всяка стъпка се грижим да е вярно $f(low) == false$ и $f(high) == true$
- Взимаме средата $mid := (low + high)/2$
- Ако $f(mid) == true$, то за следващата стъпка $high := mid$
- Ако $f(mid) == false$, то за следващата стъпка $low := mid$
- Ако low и $high$ се "срещнат", тоест $low + 1 == high$, то сме готови
- Сложността е $O(log(M) * time_f)$, където M е големината на редицата f, а $time_f$ е времето за изчисляване на една стойност на f

Двоично търсене ”по отговора”

Частни случаи

- Преди да правим каквото и да е двочно търсене, като първа стъпка може да проверим дали в редицата изобщо има и двете възможни стойности

Двоично търсене "по отговора"

Частни случаи

- Преди да правим каквото и да е двочно търсене, като първа стъпка може да проверим дали в редицата изобщо има и двете възможни стойности
  - Достатъчно е да проверим само най-левия/десния елемент на f - например ако f има стойност *true* в най-левия си елемент, то в цялата редица има само стойности *true*

Двоично търсене "по отговора"

Частни случаи

- Преди да правим каквото и да е двочно търсене, като първа стъпка може да проверим дали в редицата изобщо има и двете възможни стойности
  - Достатъчно е да проверим само най-левия/десния елемент на f - например ако f има стойност *true* в най-левия си елемент, то в цялата редица има само стойности *true*
  - Често в самите задачи е очевидно дали това е възможно и какво е решението в тези случаи

Двоично търсене "по отговора"

Код

```
int low = MIN_VAL;
int high = MAX_VAL;
while(low + 1 < high)
{
        int mid = (low + high) / 2;
        if(check(mid))
        {
                high = mid;
        }
        else
        {
                low = mid;
        }
}
```

Двоично търсене "по отговора"

Вариант с дроби

- Има и варинт с дробни параметри – тоест f да бъде дефинирана за стойности от тип `double`. Например може $f(x) == \mathit{false}$ за $x \in [0, 2.5]$ и $f(x) == \mathit{true}$ за $x \in (2.5, 5]$

Двоично търсене "по отговора"

Вариант с дроби

- Има и варинт с дробни параметри – тоест f да бъде дефинирана за стойности от тип `double`. Например може $f(x) == \mathit{false}$ за $x \in [0, 2.5]$ и $f(x) == \mathit{true}$ за $x \in (2.5, 5]$
- Тогава не се интересуваме от точната стойност на границата, ами от някакво нейно приближение (с някаква допустима грешка, най-често от порядка на 10^{-5})

Двоично търсене "по отговора"

Вариант с дроби

- Има и варинт с дробни параметри – тоест f да бъде дефинирана за стойности от тип `double`. Например може $f(x) == \mathit{false}$ за $x \in [0, 2.5]$ и $f(x) == \mathit{true}$ за $x \in (2.5, 5]$
- Тогава не се интересуваме от точната стойност на границата, ами от някакво нейно приближение (с някаква допустима грешка, най-често от порядка на 10^{-5})
- Единственото, което се променя са типовете на *low* и *high*, както и условието за спиране - има нужда да продължаваме докато разликата им не стане по-малка от нужната точност

Двоично търсене "по отговора"

Код за дробния случай

```
const double EPS = 0.00001;

double low = MIN_VAL;
double high = MAX_VAL;
while(high - low > EPS)
{
        int mid = (low + high) / 2;
        if(check(mid))
        {
                high = mid;
        }
        else
        {
                low = mid;
        }
}
```