Algorytmy i Struktury Danych

ćwiczenia

kwiecień 2020

Lista 2, zadanie 9:

Pokaż efektywną strategię gry w "10 pytań". Gra polega na odgadnięciu wartości liczby, o której wiadomo, że:

- jest dodatnia i całkowita,
- nie jest znane ograniczenie górne jej wartości.

Gracz może zadawać tylko pytania na które odpowiedź będzie TAK lub NIE. Ile pytań potrzebujesz, żeby zgadnąć dowolną liczbę (liczba pytań może zależeć od wielkości liczby)?

Rozwiązanie:

Oznaczmy szukaną liczbę jako k. Zadanie należy podzielić na dwa "podproblemy":

1. Znajdź liczbę n, taką że $n >= k$. ↪ tak naprawdę to alg podany tutaj znajduje takie n, że $k \in (\frac{n}{2}, n]$
2. Rozwiąż problem znajdowania k w skończonym przedziale $[\frac{n}{2}, n]$.

Aby rozwiązać 1. "podproblem" zaczniemy pytać czy szukana liczba jest mniejsza od n = 2. W przypadku negatywnej odpowiedzi będziemy zwiększać liczbę n dwukrotnie. Jeśli otrzymamy pozytywną odpowiedź, przejdziemy do 2. "podproblemu":

Listing 1: Pseudokod. Znajdowanie górnej granicy:

```
int znajdz_granice()
   n = 1;
   odpowiedz = NIE;

   while(odpowiedz = NIE)
      n = n*2
      cout << "Czy liczba k jest nie większa niż n?";
      cin >> odpowiedz;

return n;
```

Rozwiązanie 2. "podproblemu" polega na przeszukaniu binarnym naszego zakresu. Algorytm ten po każdym pytaniu (iteracji) zredukuje nasz zakres, gdzie znajduje się liczba k o połowę. Algorytm zakończy się, gdy nasz zakres będzie składał się z 1 elementu - będzie to szukana liczba k.

Listing 2: Pseudokod. Znajdowanie liczby k:

```
int znajdz_k(int n)
   poczatek = n/2;
   koniec = n;
   srodek = (poczatek+koniec)/2;
   odpowiedz = NIE;

   while(poczatek != koniec)
      cout << "Czy liczba k znajduje się w przedziale [poczatek,
          srodek]?";
      cin >> odpowiedz;
      if(NIE)
         poczatek = srodek + 1;
         srodek = int((poczatek+koniec)/2);
      else
         koniec = srodek;
         srodek = int((poczatek+koniec)/2);

return srodek;
```

Do wyznaczenia liczby n potrzebujemy $\log_2(n) = \lceil \log_2(k) \rceil$ zapytań. Natomiast do znalezienia liczby k w skończonym przedziale $[\frac{n}{2}, n]$ potrzebujemy w najgorszym wypadku $\log_2(n) - 1 = \lceil \log_2(k) \rceil - 1$ zapytań. Zatem całkowita złożoność naszego algorytmu wynosi:

$$\Theta(\log(n)+\log(\frac{n}{2})) = \Theta(\log(n)+\log(n)) = \Theta(2\log(n)) = \Theta(\log(n)) = \Theta(\log(k))$$

Algorytm przeszukiwania binarnego jest najszybszym sposobem na znalezienie wartości w skończonym przedziale. Dlatego też nawet gdybyśmy byli w stanie znaleźć szybszy sposób na wyznaczanie górnej granicy n, nie dałby on lepszej złożoności całego algorytmu.

Takie stwierdzenia można pisać jeśli się to udowodni!

Maksymalna liczba pytań potrzebna do odgadnięcia liczby k wynosi:

$$2 * \lceil \log_2(k) \rceil - 1$$

Aby udowodnić takie stwierdzenie trzebaby założyć np. że używamy comparison model, czyli możemy tylko porównywać liczby i dalej procedować podobnie jak w dolnym ograniczeniu dla sortowania, czyli przy użyciu drzewa decyzyjnego.