Zadania domowe. Blok 1. Zestaw 1

Maciej Poleski

3 marca 2012

1 Tablica nieskończona

Rozwiązanie składa się z dwóch częsci:

1. Znalezienie górnego ograniczenia rozmiaru tablicy
2. Klasyczny binary serach w wyznaczonym przedziale

W rozwiązaniu zakładam że ∞ jest większe od każdej liczby całkowitej oraz że $\infty \geqslant \infty$.

Faza 1

```
Wejście: A - tablica zgodnie z oznaczeniami z zadania
Wyjście: m - liczba naturalna taka że 2n > m ⩾ n

int m;
for(m=1 ; A[m]≠ ∞ ; m*=2);
```

Najpierw zauważmy że $A[m] = \infty$. Jest to warunek stopu pętli. Następnie $m \geqslant n$. Gdyby było inaczej to $A[m] \neq \infty$ a więc nie zaszedłby warunek stopu. W każdym kroku pętli m rośnie dwókrotnie oznacza to że jeżeli $A[m] = \infty$ to $A[\frac{m}{2}] \neq \infty$. Czyli $\frac{m}{2} < n$ więc $m < 2n$ i w końcu $2n > m \geqslant n$. Oznacza to że algorytm zwraca poprawny wynik pod warunkiem że się zakończy.
Zakończy się dlatego że funkcja wykładnicza 2^k jest rosnąca. A n jest skończone.
Na koniec zastanówmy się nad złożonością. m rośnie dwókrotnie przy każdym obiegu pętli. Początkowo $m = 1$, a na koniec $m < 2n$. Więc złożoność całego algorytmu wynosi $\Theta(\lg m)$. Funkcja logaryrytm binarny jest rosnąca więc $\lg m < \lg 2n$. Oznacza to że złożoność algorytmu wynosi $O(\lg 2n) = O(1 + \lg n) = O(\lg n)$

Faza 2

Dysponując obliczoną wartością m z fazy 1 natychmiast rozpoczynamy fazę 2.

```
Wejście: A - tablica zgodnie z oznaczeniami z zadania
         m - liczba uzyskana z poprzedniej fazy
         x - poszukiwana zawartość komórki
Wyjście: indeks komórki zawierającej x o ile istnieje

return binary_search(A,A+m,x)-A;
```

Algorytm `binary_search` został omówiony na wykładzie. Przykładową implementację można odnaleźć w moim rozwiązaniu zadań A i B oraz conajmniej kilku zadaniach z WdP. Oczekuję zachowania takiego jak `std::lower_bound`, czyli pierwszy argument to początek przeszukiwanego przedziału, drugi to koniec przeszukiwanego przedziału, trzeci to poszukiwana wartość. Poszukiwana wartość jeżeli istnieje to jest w tym przedziale ponieważ jest liczbą całkowitą, a zgodnie z założeniem każda liczba całkowita jest mniejsza niż ∞ oraz $A[m] = \infty$ a tablica jest posortowana niemalejąco. Wynikiem algorytmu jest pozycja, a więc po odjęciu pozycji początku przedziału uzyskujemy pozycję wewnątrz zadanego przedziału.
Złożoność algorytmu `std::lower_bound` to $O(\lg m)$. Uwzględniając fazę pierwszą złożoność obliczeniowa całego rozwiązania to $O(\lg n)$. ■