Zadania domowe. Blok 3. Zestaw 2

Maciej Poleski

6 maja 2012

1 Kiepske sortowanie

Złożoność obliczeniowa nie zależy od danych (tylko od ich wielkości). Oznaczmy $n = b - a$, a przez $f(n)$ koszt wykonania funkcji dla danych wielkości n. Rozgałęzienie w linii 4 zależy tylko od n, w linii 10 koszt rozgałęzienia jest stały. Jeżeli rozgałęzienie nie nastąpi, to samo sprawdzenie warunku jest kosztem stałym. Mamy więc dwie możliwości: dla $n \leqslant 2$ mamy $\Theta(1)$, a dla $n \geqslant 3$ koszt linii 5 jest stały, a w liniach 6, 7 i 8 przedziały mają długość $n\frac{2}{3}$ czyli koszt:

$$f(n) = 3f\left(n\frac{2}{3}\right)$$

Tą rekursje można dość łatwo rozwiązać. Wynik jest postaci 3^k gdzie k jest najmniejszą liczbą całkowitą spełniającą nierówność:

$$n\left(\frac{2}{3}\right)^k \leqslant 2$$

Po prostych przekształceniach

$$k \geqslant \log_{\frac{2}{3}} \frac{2}{n}$$

czyli

$$k = \left\lceil \log_{\frac{2}{3}} \frac{2}{n} \right\rceil$$

Czyli złożoność obliczeniowa wynosi

$$\Theta\left(3^{\log_{\frac{3}{2}} n}\right) = \Theta\left(n^{log_{\frac{3}{2}} 3}\right) \approx \Theta\left(n^{2,71}\right)$$

2 Udziwnione wyszukiwanie binarne

1. Funkcja będzie przeszukiwać liniowo tablicę od najmniejszych do największych indeksów.

2. Funkcja nie działa x może być mniejsze niż a, w efekcie zadziała linia 5 poszerzając przedział (zamiast skracając)

3. Funkcja będzie nieoptymalna. Złożoność $O\left(log_{\frac{10}{9}} n\right)$

4. Funkcja będzie bardzo nieoptymalna (liniowa). Przeszukiwanie będzie odbywać się w dwóch fazach: skakanie od końca co 10 elementów w stronę początku, następnie przeszukiwanie liniowe przedziału długości co najwyżej 10 od najmniejszych indeksów do największych.

3 Generator testów

Prześledźmy bieg wydarzeń w sytuacji gdy w pierwszej linii wybierzemy najmniejszy element tablicy (silnie mniejszy od każdego innego). W drugiej linii najmniejszy element zostanie przeniesione na początek. W pętli while pierwsza wewnętrzna pętla zakończy się natychmiast. Druga pętla wewnętrzna przejdzie całą tablicę i zatrzyma się na pierwszym elemencie (ponieważ wybrany element jest silnie mniejszy od każdego innego, ta pętla zatrzyma się dopiero na nim samym). Właśnie został dokonany nieoptymalny podział na zadanie puste i prawie pełne. Podczas generowania testów należy zwrócić uwagę na zaokrąglenie wyniku dzielenia w pierwszej linii `Sort`.

```
void generator(int *T, int l, int p)
{
    static int source=0
    int v=(l+p)/2;
    T[v]=source++;
    if(v>l) generator(T,l,v);
    if(v+1<p) generator(T,v+1,p);
}
```

Tą funkcję należy wywołać dokładnie tak samo jak funkcję `Sort`.