Lista 6 zadanie 5

Wiktor Hamberger
308982

2 czerwca 2020

Algorytm Hoare'a to prosty algorytm rozwiązujący problem selekcji. Opiera się na podobnym pomyśle co QuickSort, mianowicie na podziale zbioru na liczby mniejsze i większe od wybranego elementu. Działanie algorytmu jest następujące, powiedzmy, że dany jest zbiór **A** zawierający **n** liczb. Zadanie polega na wybraniu **k**-tej co do wielkości. Wybieramy losową liczbę **l** ze zbioru **A** i dzielimy ten zbiór na elementy mniejsze lub równe od **l** (zbiór $A_{\leq}$) oraz liczby większe od niej (zbiór$A_{>}$). Następnie, jeśli moc zbioru $A_{\leq}$ jest większa lub równa **k**, to rekurencyjnie szukamy w tym zbiorze **k**-tego elementu, w przeciwnym przypadku rekurencyjnie szukamy w zbiorze $A_{>}$ elementu $\mathbf{k}-A_{\leq}$ co do wielkości.

W najgorszym przypadku ten algorytm może mieć złożoność czasową rzędu $O(n^2)$. Dzieje się to w przypadku, gdy jako pivot będziemy wybierać w każdym kroku najmniejszy bądź największy możliwy element. Skoro wybieramy go losowo, możemy policzyć prawdopodobieństwo wybrania takiego elementu. Niech ϵ_k będzie takim zdarzeniem, że wybieramy najmniejszy bądź największy element tablicy w której pozostało k elementów. Zdarzenie ϵ oznacza natomiast najgorszy czasowo przypadek działania naszego algorytmu. W takim razie $\epsilon = \bigcap_{i=1}^{n} \epsilon_i$. Prawdopodobieństwo $P(\epsilon) = P(\bigcap_{i=1}^{n} \epsilon_i)$, a skoro ϵ_i są niezależne (dokonujemy niezależnych wyborów w każdym kroku), to możemy zredukować wyrażenie $P(\epsilon) = P(\bigcap_{i=1}^{n} \epsilon_i) = \prod_{i=1}^{n} P(\epsilon_i)$. Dla $i > 1$ $P(\epsilon_i) = \frac{2}{i}$, $P(\epsilon_1) = 1$, więc $P(\epsilon) = \prod_{i=1}^{n} P(\epsilon_i) = \prod_{i=2}^{n} \frac{2}{i} = \frac{2^{n-1}}{n!}$. Co oznacza, że ten przypadek zdarza się bardzo rzadko. Na przykład dla n=31, $2^n \approx 10^9$, a $n! \approx 8 * 10^{33}$.

Chcielibyśmy jednak udowodnić średnią złożoność tego algorytmu – $O(n)$. Podzielmy sobie kroki tego algorytmu na grupy w taki sposób, że po wykonaniu wszystkich kroków z grupy, rozmiar przeszukiwanej tablicy zmniejszył się o co najmniej 25%. To oznacza, że każda grupa kończy się w momencie, gdy w danym kroku wybierzemy pivot, który był w środkowych 50% wartości tablicy (to oznacza, że co najmniej 25% wartości było mniejszych i 25% wartości było większych od niego, więc mamy gwarancję zmniejszenia tablicy o 25%). Ponumerujmy sobie grupy zaczynając od zera. W grupie k-tej rozmiar tablicy to maksymalnie $n * (\frac{3}{4})^k$. To oznacza, że w najgorszym przypadku ostania grupa będzie miała numer $\lceil log_{4/3} n \rceil$. Niech X_k będzie zmienną losową równą liczbie wywołań rekurencyjnych w grupie k. Ilość operacji wykonywana w grupie k będzie maksymalnie wynosiła $X_k * c * n * (\frac{3}{4})^k$, dla jakiejś stałej c. W takim

razie sumaryczna liczba operacji, nazwijmy ją S, może zostać ograniczona od góry:

$$S \leq \sum_{k=0}^{\lceil log_{4/3} n \rceil} (X_k cn(\frac{3}{4})^k) = cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} (X_k(\frac{3}{4})^k).$$

Szukamy oczekiwanego czasu działania dla tablicy n-elementowej, co oznacza że szukamy wartości oczekiwanej $E[S]$, czyli $E[S] \leq E[cn \sum_{k=0}^{\lceil log_{4/3} n \rceil}(X_k(\frac{3}{4})^k)]$. Z liniowości wartości oczekiwanej:

$$E[S] \leq E[cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} (X_k(\frac{3}{4})^k)] = cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} E[X_k](\frac{3}{4})^k.$$

Z definicji $E[X_k] = \sum_{i=0}^{\infty} i * P(X_k = i)$. Gdzie $P(X_k = i)$ to prawdopodobieństwo, że pierwsze i-1 pivotów nie znajdowało się w środkowych 50% wartości, i-ty pivot już tak (zakładamy, że $P(X_k = 0) = 0$). Dla uproszczenia zakładamy, że za każdym razem wybieramy pivot ze wszystkich elementów dostępnych w tablicy na początku grupy, to tylko trochę utrudnia problem, zwiększając maksymalną możliwą ilość kroków w grupie. W takim razie $P(X_k = i) = (\frac{1}{2})^i$, więc $E[X_k] = \sum_{i=0}^{\infty} i * P(X_k = i) \leq \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$. Policzmy więc:

$$\begin{aligned}
E[S] &\leq cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} E[X_k](\frac{3}{4})^k = \\
&= cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} 2(\frac{3}{4})^k = \\
&= 2cn \sum_{k=0}^{\lceil log_{4/3} n \rceil} (\frac{3}{4})^k \leq \\
&\leq 2cn \sum_{k=0}^{\infty} (\frac{3}{4})^k = 8nc = O(n)
\end{aligned}$$

Więc algorytm Hoare'a ma oczekiwany czas działania rzędu O(n).