Grzegorz Aleksiuk 11.01.2020

Dokumentacja

AAL-3-LS urząd spraw czworakich

**Opis problemu**

Urząd spraw czworakich przyjmuje petentów w czterech różnych sprawach. Obsługa każdego petenta zajmuje 5 minut i jest wykonywana przez urzędnika specjalizującego się w danej sprawie. Urząd zatrudnia sześciu urzędników, każdy specjalizuje się w dwóch różnych sprawach. W urzędzie znajduje się jedno okienko przy którym na początku dnia siedzi urzędnik specjalizujący się w pierwszej i drugiej sprawie. Do okienka ustawia się kolejka osób z różnymi sprawami. Gdy do okienka podchodzi osoba ze sprawą inną niż ta, w której specjalizuje się urzędnik, może zostać obsłużona na dwa sposoby:

1. urzędnik przy okienku dzwoni do urzędnika specjalizującego się w tej sprawie i załatwia tą konkretną sprawę telefonicznie, przez co obsługa wydłuża się o dodatkowe 5 minut,
2. urzędnik wychodzi i woła na swoje miejsce (na trwałe) urzędnika specjalizującego się w tej sprawie, co także zajmuje dodatkowe 5 minut.

Należy znaleźć sposób obsługi petentów minimalizujący całkowity czas obsługi. Porządek ludzi w kolejce jest znany od początku i się nie zmienia.

**Metoda rozwiązania**

Przejście po kolei po kolejce petentów i sprawdzanie czy aktualny urzędnik jest w stanie obsłużyć tę osobę, jeśli tak robi to, jeśli nie to sprawdzane jest czy następną osobę może obsłużyć jeśli tak to istnieje duże prawdopodobieństwo że lepiej jest zadzwonić do tej osoby niż zmieniać urzędnika, jednocześnie sprawdzane jest czy urzędnik który potrafiłby obsłużyć co najmniej dwóch następnych petentów nie jest lepszy od aktualnego (polega na sprawdzeniu który z nich jest w stanie obsłużyć większą liczbę petentów pod rząd), jeśli jest - to zmieniamy aktualnego na kandydującego. W momencie jeżeli urzędnik nie może obsłużyć dwóch petentów pod rząd jest on zmieniany, na takiego który potrafi ich obsłużyć.

Spodziewana złożoność algorytmu jest liniowa.

**Struktury danych**

* PetitionerQueue
  + Zawiera wartość n i listę petentów
  + Solve() - zwraca najmniejszy czas potrzebny na obsłużenie petentów
  + \_\_pretenders() - porównuje dwóch urzędników i zwraca „lepszego” tzn. takiego który obsłuży większa ilość osób
  + \_\_algorithm() - algorytm wyznaczający czas obsługi kolejki petentów

**Konwencje danych**

Na wejściu podawana jest liczba n (1 <= n <= 10^6), następnie w nowej linii podawanych jest n wartości p oddzielonych spacją z zakresu (1 <= p <= 4).

Na wyjściu liczba oznaczająca ilość minut potrzebną na obsłużenie danej ilości osób.

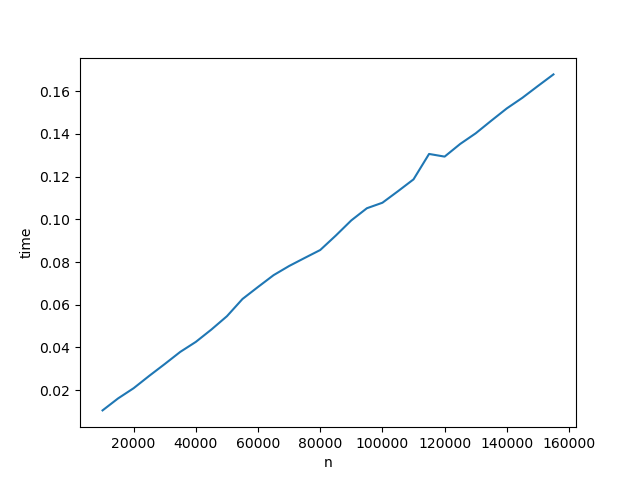
In: Out:

6 40

1 2 3 4 2 4

**Testy**

python main.py -m 3 -n 10000 -k 30 -s 5000 -r 10



|  |  |  |
| --- | --- | --- |
|  | Algorytm z asymptotą O(T(n)) | |
| n | t(n)[s] | q(n) |
| 10000 | 0.01055273 | 0.9710950533788159 |
| 15000 | 0.01616951000000002 | 0.9919790851314819 |
| 20000 | 0.02094510000000005 | 0.9637166402686644 |
| 25000 | 0.026705039999999913 | 0.9829923534207102 |
| 30000 | 0.03225556000000003 | 0.9894190021591137 |
| 35000 | 0.037950690000000085 | 0.9978115976031924 |
| 40000 | 0.042622710000000195 | 0.9805686026885929 |
| 45000 | 0.04839789000000021 | 0.98971654355418 |
| 50000 | 0.054639520000000094 | 1.0056197323534852 |
| 55000 | 0.06269810000000006 | 1.0490316855750363 |
| 60000 | 0.06833358999999994 | 1.0480449329007133 |
| 65000 | 0.07385040999999966 | 1.045529929455325 |
| 70000 | 0.07814178999999974 | 1.0272643833283759 |
| 75000 | 0.08190931999999976 | 1.0050067357309092 |
| 80000 | 0.08562062000000026 | 0.9848844866355384 |
| 85000 | 0.0923681000000002 | 1.0 |
| 90000 | 0.09949791000000055 | 1.0173452559198863 |
| 95000 | 0.10517095000000083 | 1.0187534839864745 |
| 100000 | 0.10775037000000047 | 0.9915524353104612 |
| 105000 | 0.11314155000000063 | 0.9915845250841889 |
| 110000 | 0.11869009000000119 | 0.9929301300498253 |
| 115000 | 0.13058571000000044 | 1.0449481217941665 |
| 120000 | 0.12933657999999965 | 0.9918295475746808 |
| 125000 | 0.13525902999999886 | 0.9957565479857118 |
| 130000 | 0.14024747000000062 | 0.9927698940018692 |
| 135000 | 0.14614118000000004 | 0.9961752708677224 |
| 140000 | 0.1519047499999985 | 0.998481985973191 |
| 145000 | 0.1568556899999983 | 0.9954723247677294 |
| 150000 | 0.16238645000000035 | 0.996220430357811 |
| 155000 | 0.16781713999999964 | 0.9963261579868806 |