Mateusz Wasiak 277354

Dokumentacja

Dokumentacja projektu „Urząd spraw czworakich” realizowanego w ramach przedmiotu Analiza Algorytmów(AAL).

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 tę konkretną sprawę telefonicznie, przez co obsługa wydłuża się o dodatkowe 5 minut, lub 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.

Skrócony opis problemu:

W kolejce są zadania 4 różnych typów. W urzędzie jest 6 pracowników specjalizujących się w dwóch różnych zadaniach. Jeżeli przy okienku jest urzędnik, który specjalizuje się w danym zadaniu to wykonuje zadanie w 5 minut. Jeśli nie to może wykonać je w 10 minut, lub zawołać kogoś innego na swoje miejsce – trwa to 10 minut (razem z obsługą aktualnego zadania przez zawołanego pracownika). Należy uszeregować pracowników tak, by czas obsługi był jak najmniejszy.

Metody rozwiązania zadanego problemu:

Metoda Brutalna – wykorzystywana głównie do testowania następnej metody :
Metoda ta sprawdza czy dane zadanie może być wykonane przez aktualnie pracującego pracownika. Jeśli tak to pracownik wykonuje to zadanie, a czas wykonania zadań zwiększa się o 5. W przeciwnym wypadku uruchamiamy cztery razy rekurencyjnie tą samą metodę tylko z innymi zadaniami optymalnie wykonywanymi przechodząc już do następnego zadania. Sprawdzamy jaki czas wykonania jest najmniejszy i takie uszeregowanie pracowników jest optymalne. Czas zwiększa się o 10 (ponieważ albo zawołaliśmy innego pracownika lub wykonaliśmy to zadanie samodzielnie).

Alternatywna metoda – wykorzystywana jako główny algorytm do szeregowania pracowników i obliczania optymalnego czasu obsługi.
Metoda ta najpierw sprawdza czy aktualne zadanie jest obsługiwane optymalnie. Jeśli tak to zwiększa ona czas o 5, nie zmienia pracowników i przechodzi do następnego zadania. Następnie jesteśmy w przypadku takim, że pierwsze zadanie nie jest optymalne, więc sprawdzamy czy następne zadanie też nie jest optymalne. Jeśli również nie jest, to zmieniamy pracownika przed wykonaniem pierwszego zadania. Zwiększamy czas o 10 i sprawdzamy czy następne zadania są takie same jak pierwsze zadanie by szybciej obliczyć czas potrzebny do obsługi tych zadań. Trzecim krokiem jest sprawdzenie następnych zadań gdy pierwsze zadanie nie jest optymalne, a drugie jest optymalne. Najpierw dodamy do czasu 15 bo nie ważne czy zawołamy czy nie te dwa zadania które już rozważyliśmy wykonają się w 15 minut. Należy sprawdzić tu kilka różnych wariantów zadań w kolejce. Jeżeli występują zadania, które są takie same jak to na drrugim miejscu to iterujemy po nich aż nie znajdziemy innego. Zwiększając odpowiednio czas o 5 za każde

zadanie. Następnie w zależności od tego jakie inne zadanie zostało po tym iterowaniu znalezione wykonujemy różne czynności. Jeśli jest to ta samo zadanie co pierwsze to zmieniamy pracownika. Jeśli jest to zadanie, które jest innym zadaniem obsługiwanym przez danego pracownika optymalnie to nie zmieniamy pracowników. Jeżeli jest to inne zadanie – czyli takie, które nie było na początku rozważane i nie jest zadaniem rozwiązywanym optymalnie przez danego pracownika to należy sprawdzić następne zadania. Jeżeli znów znajdziemy zadanie, które było 2 w kolejce to iterujemy po nim analogicznie do jak w poprzednim kroku. W momencie gdy znajdziemy inne zadanie to : Jeśli jest to to samo zadanie co pierwsze w kolejce to zmieniamy pracownika. W przeciwnym wypadku nie zmieniamy pracowników.

Złożoność **O(n)**

Metoda programowania dynamicznego – Wykorzystanie metod programowania dynamicznego do znalezienia odpowiedniego rozwiązania. Metoda ta zakłada zbudowanie skierowanego warstwowego grafu którego wierzchołki odpowiadałyby pracownikom w urzędzie, natomiast krawędzie przepływowi tych pracowników przy okienku. Wagi krawędzi odpowiadają temu jaki jest czas trwania obsługi danego petenta. Po zbudowaniu grafu używany jest algorytm programowania dynamicznego do znalezienia najkrótszej ścieżki w tym grafie. Struktura danych reprezentująca graf, która została wykorzystana w tym przypadku to wektor którego elementami są hashmapy. Każde element wektora – hashmapa odpowiada jednemu wierzchołkowi grafu, natomiast w hashmapie przechowywane są mozliwe

Złożoność **O(n)**

Zrzuty ekranu tabeli porównującej złożoność teoretyczną i pawdziwą (programowanie dynamiczne, alternatywny algorytm) **:**

n	t(n)[ms]	q(n)
10000	118.900	0.900758
20000	246.290	0.932917
30000	376.929	0.951841
40000	505.293	0.956994
50000	631.129	0.956257
60000	763.013	0.963400
70000	894.701	0.968291
80000	1019.470	0.965407
90000	1158.147	0.974871
100000	1277.215	0.967587
110000	1481.521	1.020332
120000	1603.452	1.012280
130000	1858.845	1.083243
140000	1938.385	1.048909
150000	2020.938	1.020676
160000	2108.194	0.998198
170000	2227.919	0.992834
180000	2357.492	0.992210
190000	2479.249	0.988536
200000	2691.525	1.019517
210000	2935.452	1.058966
220000	2959.845	1.019230
230000	3060.285	1.007999
240000	3164.028	0.998746
250000	3302.403	1.000728
260000	3542.840	1.032296
270000	3686.784	1.034451
280000	3864.078	1.045476
290000	3892.708	1.016904
300000	3912.071	0.987897
310000	4101.107	1.002226
320000	4193.011	0.992664
330000	4313.901	0.990335
340000	4460.690	0.993915
350000	4577.769	0.990859
360000	4896.377	1.030382
370000	5408.338	1.107358
380000	5707.034	1.137766
390000	5724.403	1.111966
400000	6008.340	1.137943
410000	5893.734	1.089012
420000	5931.173	1.069836
430000	6073.217	1.069982
440000	6224.222	1.071664
450000	6361.822	1.071014
460000	6538.982	1.076907
470000	6638.498	1.070035
480000	6778.450	1.069831

n	t(n)[ms]	q(n)
480000	6778.450	1.069831
490000	6846.645	1.058541
500000	6561.364	0.994146
510000	7272.136	1.080234
520000	7594.614	1.106441
530000	7678.661	1.097579
540000	7803.366	1.094748
550000	7870.937	1.084151
560000	7366.594	0.996563
570000	8118.959	1.079075
580000	8335.096	1.088701
590000	8492.910	1.090512
600000	8683.591	1.096413
610000	8806.259	1.093674
620000	8855.826	1.082090
630000	8350.883	1.004195
640000	8367.588	0.990482
650000	8526.659	0.993783
660000	8729.866	1.002051
670000	8854.787	1.001220
680000	8814.579	0.982016
690000	8959.658	0.983713
700000	9112.166	0.986165
710000	9242.517	0.986184
720000	9366.852	0.985569
730000	9627.785	0.999147
740000	9590.079	0.981785
750000	9728.708	0.982698

n	t(n)	q(n)
1000000	35.000	0.875000
2000000	73.600	0.920000
3000000	113.160	0.943000
4000000	152.616	0.953850
5000000	191.862	0.959308
6000000	233.886	0.974526
7000000	271.389	0.969245
8000000	310.939	0.971684
9000000	349.294	0.970261
10000000	396.029	0.990073
11000000	449.403	1.021370
12000000	488.840	1.018417
13000000	537.284	1.033239
14000000	560.228	1.000408
15000000	629.823	1.049705
16000000	650.082	1.015754
17000000	679.308	0.998983
18000000	709.931	0.986015
19000000	746.293	0.981965
20000000	804.429	1.005537
21000000	877.343	1.044456
22000000	894.534	1.016516
23000000	940.453	1.022232
24000000	984.345	1.025360
25000000	993.535	0.993535
26000000	1025.953	0.986494
27000000	1075.795	0.996107
28000000	1129.080	1.008107
29000000	1161.808	1.001559
30000000	1179.781	0.983151
31000000	1218.778	0.982886
32000000	1263.278	0.986936
33000000	1305.428	0.988960
34000000	1419.443	1.043708
35000000	1448.644	1.034746
36000000	1430.564	0.993448
37000000	1469.356	0.992808
38000000	1505.236	0.990287
39000000	1545.424	0.990656
40000000	1587.542	0.992214
41000000	1626.054	0.991496
42000000	1667.105	0.992325
43000000	1708.511	0.993320
44000000	1779.151	1.010881
45000000	1789.815	0.994342
46000000	1850.982	1.005968
47000000	1874.498	0.997073
48000000	1918.750	0.999349
49000000	1942.375	0.991008
50000000	1990.537	0.995269
51000000	2021.654	0.991007
52000000	2067.965	0.994214
53000000	2166.997	1.022168
54000000	2228.800	1.031852
55000000	2227.380	1.012445
56000000	2242.538	1.001133
57000000	2273.654	0.997217
58000000	2319.965	0.999985
59000000	2343.497	0.993007
60000000	2382.850	0.992854
61000000	2423.085	0.993068
62000000	2518.608	1.015568
63000000	2549.861	1.011850
64000000	2606.186	1.018041
65000000	2634.119	1.013123
66000000	2670.612	1.011595
67000000	2720.961	1.015284
68000000	2757.996	1.013969
69000000	2795.500	1.012862
70000000	2843.450	1.015518
71000000	2861.245	1.007481
72000000	2910.824	1.010703
73000000	2939.682	1.006741
74000000	2983.568	1.007962
75000000	3059.257	1.019752
76000000	3082.326	1.013923
77000000	3117.133	1.012056
78000000	3154.613	1.011094
79000000	3136.161	0.992456
80000000	3177.216	0.992880
81000000	3261.322	1.006581
82000000	3262.132	0.994552
83000000	3306.113	0.995817
84000000	3348.911	0.996700
85000000	3412.691	1.003733
86000000	3458.569	1.005398
87000000	3498.057	1.005189
88000000	3516.106	0.998894
89000000	3553.111	0.998065
90000000	3674.211	1.020614
91000000	3681.921	1.011517
92000000	3750.192	1.019074
93000000	3826.919	1.028742
94000000	3785.692	1.006833
95000000	3822.069	1.005808
96000000	3824.307	0.995913
97000000	3867.831	0.996864
98000000	4030.083	1.028082
99000000	4006.108	1.011644