Sprawozdanie PTSZ nr 1

Kamil Kowalczyk
136742, i2, L1
zajęcia środa 11:45

Generator instancji

Generator został zaprojektowany z użyciem maszyny generującej liczby pseudolosowe:

$$p_j = random(10,\ 50)$$
$$r_j = j + random(1, 20)$$
$$d_j = r_j + p_j + random(0, 10)$$
$$w_j = random(1,\ 10)$$

gdzie $random(n,\ m)$ generuje liczbę całkowitą z przedziału [n, m-1]

Z uwagi na sposób generowania oczekiwanego czasu zakończenia, z niewielkim zapasem "luzu" czasowego (od 0 do 9), instancje generowane przez ten program są ciężkie do rozwiązania i posiadają wiele konfliktów czasowych nachodzących na siebie zadań.

Algorytm

Bazuje na tym, że spóźnione zadania dostają równą karę, bez względu na to, jak długie było to opóźnienie. Zastosowano, więc mechanizm przenoszenia zadań, które są spóźnione i zarazem nie są pożądane (mają niską wagę w_j) na koniec listy. Dzięki temu czas, który musielibyśmy poświęcić na wykonanie tego zadania, zaoszczędzimy na rzecz następnych zadań.

Wpierw wszystkie zadania są sortowane rosnąco po czasach gotowości p_j. Następnie program tworzy dwie listy *doneJobs* i *behindJobs*. W pierwszej będą przechowywane zadania, które zostały wybrane jako te, które mają być wykonane. Druga lista reprezentuje zadania, które zdecydowano, że będą miały opóźnienie, więc mogą zostać wykonane na końcu.
Program dodaje kolejne zadania do listy *doneJobs* do momentu, w którym któreś zadanie T_j nie zdąży się wykonać. Algorytm wykrywa taką sytuację i sprawdza, czy gdyby porzucił poprzednie zadanie T_{j-1} i przeniósł je na koniec listy wynikowej, to czy to spóźnione T_j

zdążyło by się wtedy wykonać. Jeżeli tak, sprawdzany jest jeszcze dodatkowy warunek - czy waga obecnego zadania T_j jest większa od poprzedniego T_{j-1} . Jeżeli oba te warunki są spełnione, następuje zamiana - ostatnie dodane do listy *doneJobs* zadanie T_{j-1} zostaje ściągnięte i wrzucone do listy *behindJobs*, a nowe zadanie T_j ląduje na jego miejscu w liście doneJobs. W przeciwnym przypadku nie dokonywana jest żadna zamiana i zadanie T_j wstawiane jest do listy *behindJobs*.
Algorytm kończy działąnie, gdy przejdzie po wszystkich zadaniach z początkowej listy. Ostatnim etapem programu jest zwrócenie listy, która jest wynikiem konkatenacji kolejno listy *doneJobs* i *behindJobs*.

Złożoność:

Sortowanie listy po kluczu (python, funkcja sort): $O(n\ log\ n)$
źródło: https://wiki.python.org/moin/TimeComplexity

Przejście po całej liście i dokonywanie zamian: $O(n)$

Wyniki

n	Wartość kryterium dla sztucznego pliku wynikowego	Wartość kryterium uzyskana przez własny algorytm	Względna różnica w [%]	Czas działania algorytmu [s]
50	**266**	**262**	**1,50**	**0,238**
100	**523**	**518**	**0,96**	**0,234**
150	**782**	**768**	**1,79**	**0,233**
200	**1136**	**1109**	**2,38**	**0,235**
250	**1341**	**1303**	**2,83**	**0,245**
300	**1627**	**1615**	**0,74**	**0,233**
350	**1899**	**1836**	**3,32**	**0,236**
400	**2130**	**2093**	**1,74**	**0,237**
450	**2587**	**2555**	**1,24**	**0,236**
500	**2635**	**2566**	**2,62**	**0,235**

Średnia względna różnica dla całej biblioteki wynosi [%]: 42,37