Politechnika Wrocławska

Komputerowo Zintegrowane Wytwarzanie

Ćwiczenie 2

Hanna Grzebieluch, 264209
Termin zajęć: Środa, $13^{15} - 15^{00}$
Termin oddania zadania: 09.04.2024
Spóźnienie: brak
Sugerowana ocena: 3,5

Wydział Informatyki i Telekomunikacji
Informatyczne Systemy Automatyki

10 kwietnia 2024

1 Wstęp

Algorytm WiTi jest algorytmem ściśle związanym z szeregowaniem zadań. Dane jest n zadań do wykonania, gdzie jednocześnie na maszynie może być wykonywane tylko jedno. Każde z zadań jest opisane trzema parametrami:

1. p_i - czas trwania,
2. w_i - waga,
3. d_i - pożądany termin zakończenia

Jeśli dane zadanie jest spóźnione naliczana jest kara (oparta na wadze zadania). Szukane jest uszeregowanie o minimalnej sumie kar. Spóźnienie wyznaczamy na podstawie wzoru

$$T_i = \begin{cases} 0 & \text{dla } C_i \leqslant d_i; \\ C_i - d_i & \text{dla } C_i > d_i \end{cases}$$

gdzie C_i to czas zakończenia zadania i-tego.
Minimalizujemy sumę ważonych spóźnień: $\sum_i w_i T_i$, gdzie T - czas spóźnienia.

1.1 Programowanie dynamiczne - teoretyczny opis algorytmu

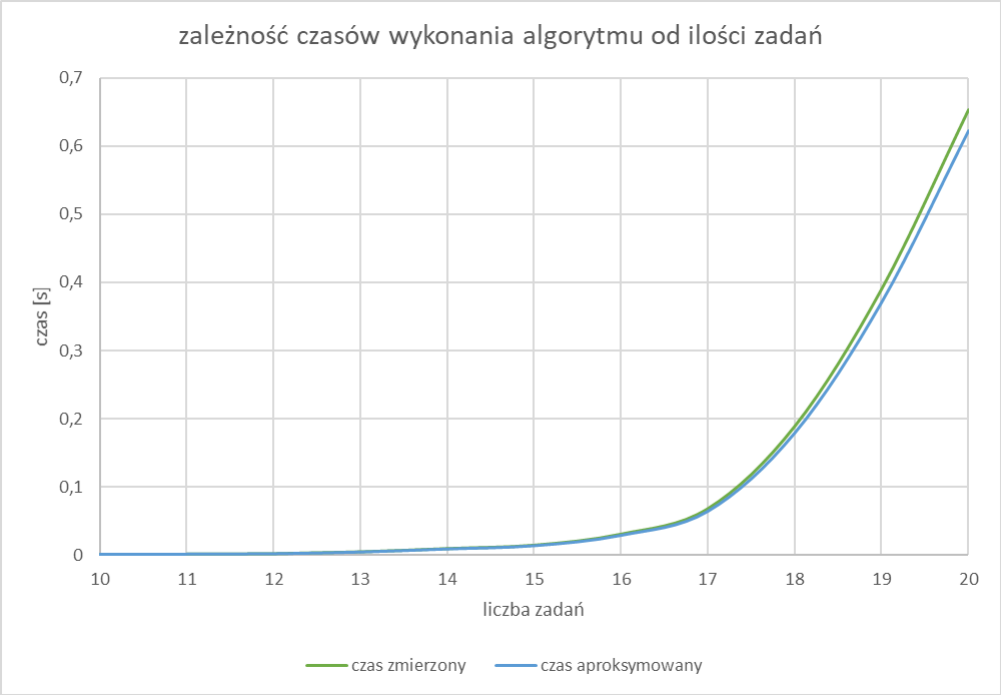
Programowanie dynamiczne jest techniką projektowania algorytmów, która polega na rozwiązywaniu podproblemów i zapamiętywaniu ich wyników.
Algorytm programowania dynamicznego rozwiązuje podproblemy od najmniejszych do największych i zapisuje optymalne wartości w tablicy. Dzięki temu podejściu każdy podproblem rozwiązywany jest tylko jeden raz, co z kolei pozwala na znaczne zaoszczędzenie czasu i pamięci.

W tym celu w programie została stworzona tablica dynamiczna F o rozmiarze 2^n, której elementami są długości częściowych uszeregowań. Ponadto każdy z podzbiorów otrzymuje "etykietę", która jest w postaci liczby binarnej i umożliwia jego jednoznaczną identyfikację.
Sam algorytm przeprowadza rekurencję po podproblemach i jednocześnie korzysta z faktu, że znamy czas zakończenia ostatniego zadania. Korzystamy z zależności przedtsawionej na wykładzie:

$$F(I) = min_{k \in I}\{F(I \setminus \{k\}) + K_k(C(I))\}$$

gdzie I - zbiór zadań F(I) - kara optymalnego uszeregowania zadań I, $K_k(t)$ - kara zadania k zakończonego w czasie t i C(I) - długość uszeregowania zadań I.

2 Porównanie czasów



Rysunek 1: Wykres porównujący czas aproksymowany i zmierzony

Wykres został stworzony na podstawie tabelki, w której znajdują się zmierzone czasy działania algorytmu oraz obliczone czasy estymowane dla różnych ilości zadań.

liczba zadań	czas algorytmu [ms]	czas algorytmu [s]	czas aproksymowany	2^n
10	0,28	0,00028	0,00025	1024
11	0,69	0,00069	0,00063	2048
12	1,38	0,00138	0,00127	4096
13	4,12	0,00412	0,00383	8192
14	8,87	0,00887	0,00828	16384
15	13,93	0,01393	0,01306	32768
16	30,13	0,03013	0,02836	65536
17	67,48	0,06748	0,06373	131072
18	188,56	0,18856	0,17864	262144
19	390,15	0,39015	0,37064	524288
20	653,34	0,65334	0,62223	1048576

Rysunek 2: Tabela wartości

Im większa liczba zadań do uszeregowania, tym większa rozbieżność w czasie zmierzonym oraz aproksymowanym. Może to wynikać to z pewnej niedokładności pomiaru czasu wykonania algorytmu.

3 Uzupełnienie

W ramach uzupełnienia do kodu źródłowego dodana została możliwość wyświetlenia wynikowej permutacji.

```
dane.10:
czas optymalny: 766
kolejnosc: 6 9 2 5 1 3 4 7 8 10
total time: 29 [ms]
```

Rysunek 3: Przykładowy wynik dla pliku data.10