**Projektowanie Efektywnych Algorytmów**

Jednoprocesorowy problem szeregowania zadań

Metoda programowania dynamicznego

Autor:

Nr indeksu:

Termin oddania:

Prowadzący:

Termin zajęć:

Kierunek: Informatyka

1. **Wstęp teoretyczny**

Pełna nazwa problemu to „Jednoprocesorowy problem szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań”.

Danych jest *n* zadań, które mają być wykonane bez przerwań przez pojedynczy procesor, mogący wykonywać co najwyżej jedno zadanie jednocześnie. Każde zadanie jest dostępne do wykonania w chwili zero, do wykonania wymaga pi > 0 jednostek czasu, ma określoną wagę wi > 0 i ocze-kiwany termin zakończenia wykonywania di > 0. Zadanie *i* jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie d*i*, miarą tego opóźnienia jest Ti = max(0, Ci – di), gdzie Ci to termin zakończenia zadania *i*. Problem polega na znalezieniu takiej kolejności zadań aby zminimalizować kryterium .

1. **Sposób przeprowadzenia eksperymentu**
   1. **Algorytm**

Algorytm który został użyty w rozwiązaniu problem to algorytm programowania dynamicznego (DPA). Problem został podzielony na 2n podzbiorów. Zaczynając od pojedynczych zadań, a kończąc na wszystkich razem, każde następne wyliczenie opóźnienia korzystało z wyliczonych już wcześniej opóźnień dla mniejszych podzbiorów. Algorytm dynamicznie wybiera najlepszy podzbiór i na jego podstawie wylicza opóźnienie dla aktualnego. Opóźnienie z poprzednich podzbiorów które było złożone z i-1 zadań jest dodawane do opóźnienia kolejno każdego zadania (dla danego maksymalnego czasu sumy połączonych w danej chwili zadań), które nie było zawarte w poprzednich podzbiorach. Pośród tych sum opóźnień wybierany jest czas minimalny, który był zapamiętywany dla tego podzbioru i wykorzystany przez następne do wyliczenia następnych sum opóźnień.

Wskazanie kolejności zadań jest badane na podstawie już gotowego zbioru wszystkich podzbiorów, które sprawdza wstecz każdy kolejny podzbiór z którego korzystał i które zadanie było na końcu.

Dodatkowo do programu dołączyłem algorytm przeglądu zupełnego (Brute Force), aby mieć z czym porównać algorytm programowania dynamicznego. Przegląd zupełny polega na sprawdzeniu absolutnie wszystkich kombinacji po kolei i wybraniu najlepszego rozwiązania. Jest to algorytm najbardziej złożony czasowo.

* 1. **Założenia**
     1. Program testowy jest wykonany jako aplikacja konsolowa
     2. Algorytm DPA wymaga dość dużego zasobu pamięci, ponieważ musi on wygenerować 2n par <podzbiór, opóźnienie dla danego podzbioru>. Ze względu na ograniczenia sprzętowe, maksymalna instancja problemu zawiera 24 zadania (powyżej 24 program nie wykonuje prawidłowo algorytmu)
     3. Badania zostały wykonane dla 7 różnych wartości N zadań
     4. Dla każdej wartości N wygenerowano 50 losowych instancji problemu (w sprawozdaniu podano tylko wyniki uśrednione)
     5. Implementacja została wykonano obiektowo
     6. Do programu dołączono algorytm przeglądu zupełnego, testy dla niego zostały wykonane na podobnych zasadach jak dla DPA oprócz:
        1. Ilość losowych instancji na każde N to 10
        2. Dla niektórych N nie wykonano testów ze względu na długi czas wykonywania

1. **Wyniki**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość zadań [n] | 6 | 9 | 12 | 15 | 18 | 21 | 24 | Algorytm |
| Średni czas wykonania [ms] | 0,0809655 | 0,886956 | 8,94367 | 91,4503 | 920,358 | 9260,12 | 89113,9 | DPA |
| 0,0288511 | 11,9369 | 19927,5 | N/A | N/A | N/A | N/A | Brute Force |

1. **Wnioski**

Algorytm programowania dynamicznego jest dość efektywnym algorytmem, wyniki można osiągnąć nieporównywalnie szybciej niż w przypadku przeglądu zupełnego. Jego główną wadą jest wymóg zapamiętania 2n podzbiorów w pamięci komputera razem z wyliczonym dla każdego z nich wynikiem. Jest to bardzo wymagające i problemem już bardziej nie jest czas wykonania, tylko jego wymagania co do wolnej pamięci komputera. Czas wykonania rośnie wykładniczo ale nie tak szybko jak przegląd zupełny. Złożoność obliczeniowa tego algorytmu to suma operacji przejścia przez wszystkie podzbiory, wyliczenia całkowitego czasu dla danego podzbioru i wyliczenia opóźnienia dla danego podzbioru jak suma opóźnień wszystkich zadań.