Piotr Giedziun 184731 Termin: Pn/TP - 15:15

Projektowanie efektywnych algorytmów

Labolatorium

Problem szeregowania zadań - B&B

Prowadzący:
mgr. inż. Karolina Mokrzysz

1. **Opis problemu**

Problem szeregowania zadań jest problemem optymalizacji dyskretnej, występuje w systemach czasu rzeczywistego. Problem ten rozwiązywany jest przez algorytm szeregowania, który decyduje o tym jak rodzielić czas dostępu do procesora.

Podstawowymi terminami są:
- zadanie - zbiór operacji, które należy wykonać.
- procesor - obiekt wykonujący operacje.

Istnieje wiele klasyfikacji szeregowania zadań, najbardziej istotne to sposoby szeregowania zadań, ilość procesorów, kryterium itp.

Sposoby szeregowania zadań:
- szeregowanie statyczne - kolejność oraz rodzaj wykonywancych zadań jest określany przed rozpoczęciem procesu.
- szeregowanie dynamiczne - spowodowane istnienim elementów stochastyczych.

Podział ze względu na ilość procesorów:
- system jednoprocesorowy - wszystkie zadania są wykonywane przez jeden procesor
- system wieloprocesorowy - każdy procesro może przetwarzać dostępne zadania.

Istnieje wiele metod rozwiązywania problemu szeregowania, najbardziej popularne to:
- wielomianowe algorytmy dokadne - tylko dla problemów klasy P (rozwiazania optymalne w czasie wielomianowym)
- programowanie dynamiczne - efektem jest otrzymanie rozwiazania optymalnego. Przykładową implementacją jest backtracking (procedura poszukiwania wstecznego)
- metoda podziału i ograniczeń - trwa tak długo, aż dokonany będzie przegląd wszystkich węzłów. Efektem jest wynik dokładny.
- schematy aproksymacyjne - efektem jest wynik przybliżony.

W naszym przypadku problem jest prymitywny.
- zadania dostępne do wykonania są od początku (chwili zero)
- mamy do czynienia z szeregowaniem statycznym
- zadanie rozpatrywane jest na systemie jednoprocesorowym

Podstawowa wersja programu bazuje na przeglądzie zupełnym.
Dodatkowo zaimplementowałem odcinanie gałęzi dla której rozwiązanie częściowe jest większe niż dotychczas znalezione rozwiązanie całkowite.

2. Implementacja

Program został zaimplementowany w języku Python, wyróżnić można dwie podstawowe warstwy:

- Model/Kontroller - zaimplementowany obiektowo, na poziomie abstrakcji zadań oraz Procesora
- Widok - Interfejs użytkownika - zaimplementowany w TornadoWeb (webbrowser base) - wyświetla wyniki w formie grafu

3. Przykład działania

Program jest obsługiwany przez interfejs graficzny (GUI), dostępne poprzez przeglądarkę. Program hostuje lokalny serwer plików statycznych oraz dynamiczne API, poprzez które zwraca wynik działania. Standardowo serwer lokalny postawiony zostanie na porcie 8888.

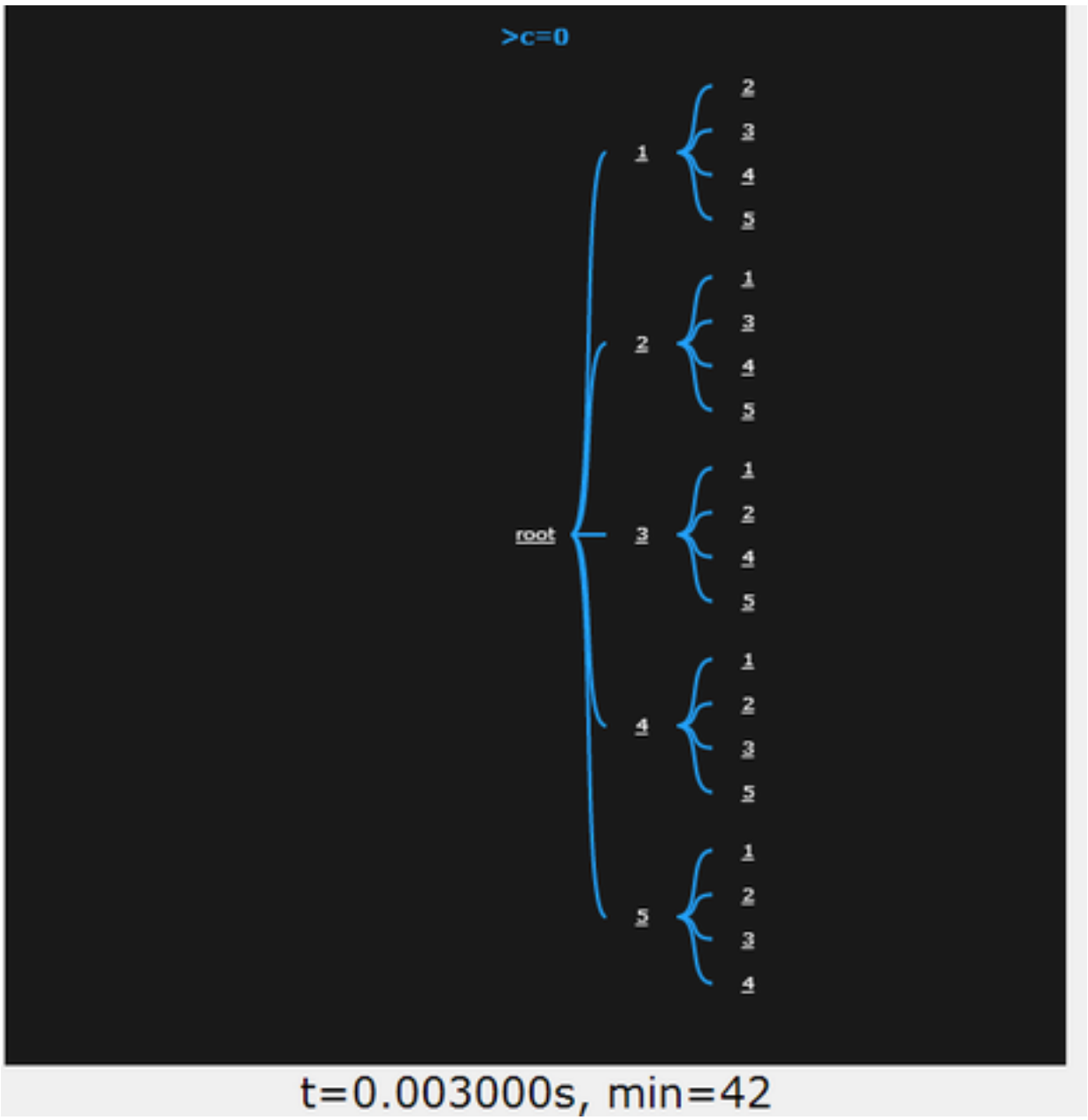
Wymagania programu:

- Program działa na Windows/Mac/Linux
- Python w wersji 2.7.x
- Tornado 2.4.1 (http://www.tornadoweb.org/)
- Przeglądarka z obsługą javascript
- Wolny port 8888 (nie są wymagane uprawnienia administracyjny do zbindowania)

Program należy uruchomić z poziomu linii poleceń (terminal) poleceniem
> python app.py

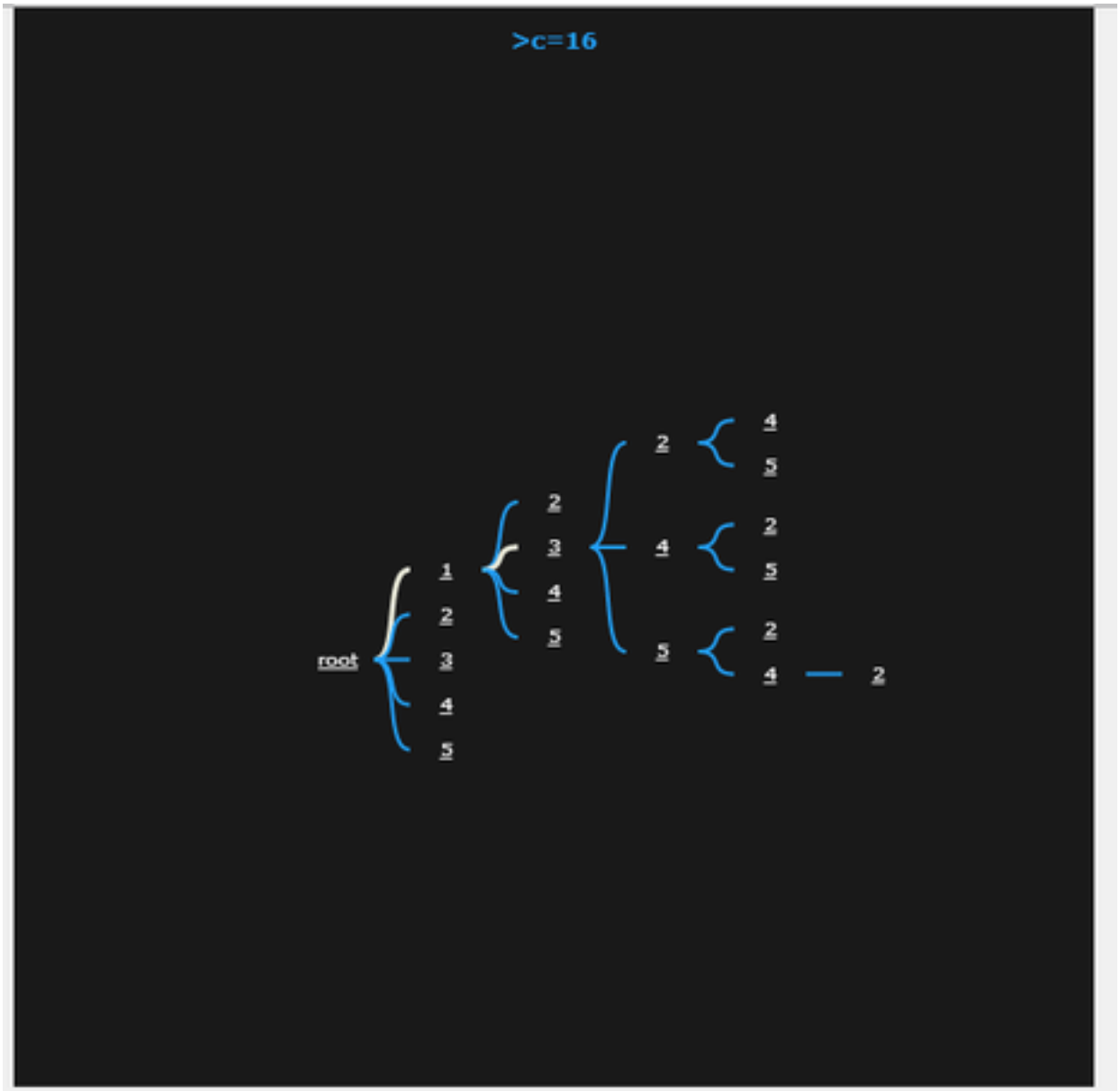
Następnie w przeglądarce wejść na adres
http://127.0.0.1:8888/

Otrzymamy listę dostępnych zadań. Uruchomienie zadania odbywa się poprzez kliknięcie w odpowiedni link.



Po wybraniu zestawu otrzymamy interaktywne drzewo, które pozwala na przeglądanie jego zawartości. Po wejściu w odpowiedni węzeł na górze ekranu wyświetlą się informacje o węźle.

Na dole ekranu widzimy czas wykonywania skryptu oraz jego wynik.



Przykład rozwiniętego drzewa.

4. **Wyniki**

Wersja 1 - przegląd zupełny

Wersja 2 - odcinanie gałęzi dla której rozwiązanie częściowe jest większe niż dotychczas znalezione rozwiązanie całkowite

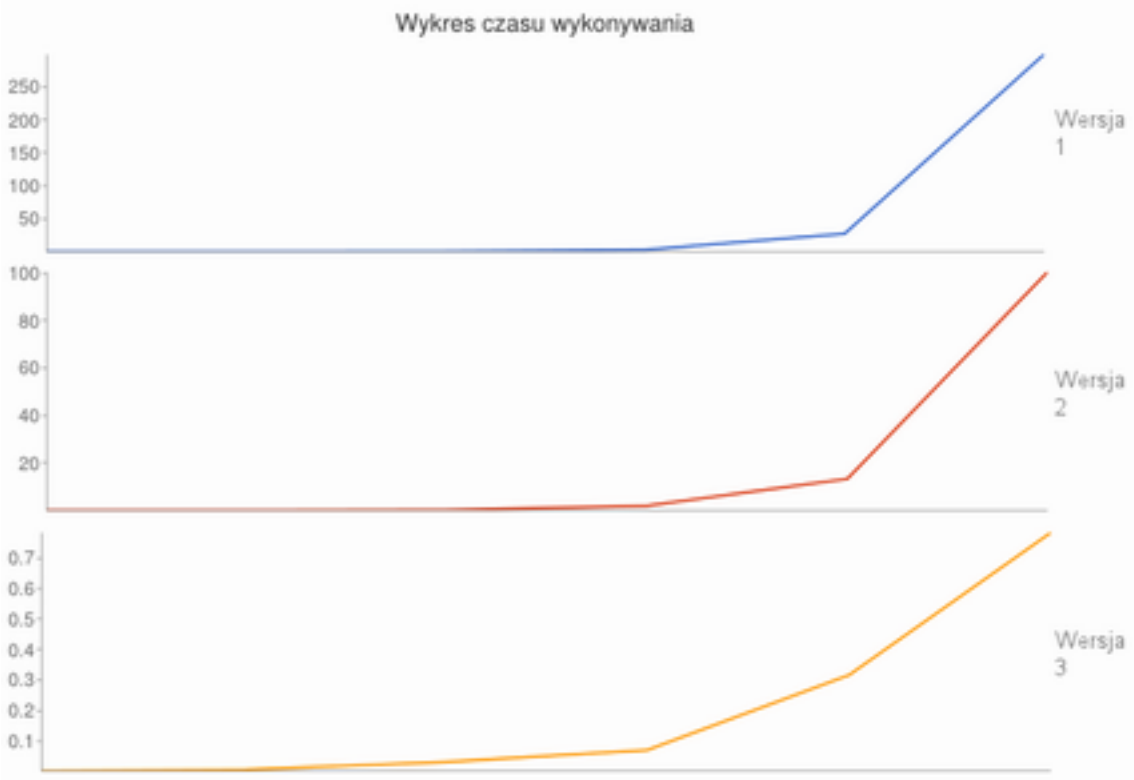
Wersja 3 - dla każdego węzła sprawdzamy pewną własność. Zakładamy, że na liście uporządkowanej będziemy mieli elementy A B C D E. Znamy również wartość kryterium dla takiej permutacji (totalCost). Robimy następujący eksperyment. Zamieniamy element ostatni z przedostatnim, liczymy kryterium (dla otrzymanej

Licz. Zad	**5**	**6**	**8**	**9**	**10**	**11**
Wersja 1	0.001s	0.005s	0.301s	2.676s	26.961s	299.167s
Wersja 2	0.001s	0.005s	0.196s	1.936s	13.255s	100.439s
Wersja 3	0.002s	0.006s	0.031s	0.070s	00.316s	000.783s
Wynik (TWT)	42	82	223	313	370	454

Uzyskane wyniki

- **5**
  - **[5, 2, 1, 4, 3]**
- **6**
  - **[5, 6, 2, 4, 1, 3]**
- **8**
  - **[7, 5, 6, 4, 2, 8, 1, 3]**
- **9**
  - **[7, 5, 6, 4, 9, 2, 8, 1, 3]**
- **10**
  - **[7, 5, 6, 14, 9, 2, 8, 1, 3]**
- **11**
  - **[7, 5, 6, 14, 9, 11, 2, 8, 1, 3]**

Wykres 1 - przedstawia kolejne zadania w czasie



Wykres 2 - przedstawia krzywe kolejnych wersji

5. **Wnioski**
    a. Jak widać z wykresu pierwszego każda kolejna wersja działa znacznie szybciej.
    b. Krzywe wykresów są podobne (wykres drugi). Algorytmy wykonywane są z tym samym stopniem złożenia (przycinanie) skraca czas liniowo
    c. Otrzymane wyniki były poprawne