Lista 2 - AiSD

Zadanie: 7
Prowadzący: Paweł Gawrychowski

Wiktor Pilarczyk

29 marca 2020

1 Wprowadzenie

W zadaniu mamy podane n zadań, wraz z czasami a_i i b_i - czas wykonania zadania odpowiednio na komputerze A i B. Zadanie musi najpierw zostać wykonane na komputerze A, a następnie na B (komputery mogą działać równolegle). Chcemy zminimalizować czas wykonania ostatniego zadania na komputerze B.

2 Własności zadania

Oczywiste jest, że jesli możemy wybrać zadanie do wykonania na komputerze B to nie czekamy tylko wykonujemy dowolne z nich, a także przy wielu dostępnych zadaniach (ich część na komputerze A została zakończona) nie ma wpływu, które będziemy wykonywać w danym momencie, ponieważ i tak liczy się kiedy zostanie wykonane ostatnie zadanie, więc dla uproszczenia niech zadania wykonane na komputerze B będą w tej samej kolejności jak zadania na komputerze A.

Stwórzmy funkcję $F(a_1, ..., a_n, b_1, ..., b_n)$ która dla zadanych ciągów obliczy czas wykonania ostatniego zadania na komputerze B.

$F(a_1, ..., a_n, b_1, ..., b_n) = \sum_{i=1}^{n}(b_i + t_i) = \sum_{i=1}^{n} b_i + \sum_{i=1}^{n} t_i$, gdzie t_i oznacza długość przerwy pomiędzy wykonaniem zadania b_{i-1}, a rozpoczęciem wykonania zadania b_i (czas bezczynności komputer B, więc w przypadku zadania b_1 jest to czas wykonanywania zadania a_1). Na $\sum_{i=1}^{n} b_i$ nie mamy wpływu więc chcemy zminimalizować $\sum_{i=1}^{n} t_i$.

$t_i = max(0, \sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1}(b_j + t_j))$, jest to maksimum z różnicy pomiędzy możliwym czasem rozpoczęcia zadania b_i, a czasem zakończenia zadania b_{i-1}, a 0, ponieważ jeśli czas zakończenia zadania b_{i-1} jest większy niż zakończenia zadania a_i to nie musimy czekać więc $t_i = 0$.

Teza: $\sum_{i=1}^{n} t_i = max_{i=1}^{n}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j)$

Dowód. Indukcyjnie po n.

Baza: $n = 1$
Wtedy

$$t_1 = a_1 = max(a_1)$$

spełnia tezę.

Założenie indukcyjne: $\sum_{i=1}^{n} t_i = max_{i=1}^{n}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j)$
Teza indukcyjna: $\sum_{i=1}^{n+1} t_i = max_{i=1}^{n+1}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j)$

$$\sum_{i=1}^{n+1} t_i = \sum_{i=1}^{n} t_i + t_{n+1} =^{z\ def} \sum_{i=1}^{n} t_i + max(0, \sum_{j=1}^{n+1} a_j - \sum_{j=1}^{n}(b_j + t_j)) =$$

$$max(\sum_{i=1}^{n} t_i, \sum_{j=1}^{n+1} a_j - \sum_{j=1}^{n} b_j) =^{ZI} max(max_{i=1}^{n}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j), \sum_{j=1}^{n+1} a_j - \sum_{j=1}^{n} b_j) =$$

$$max_{i=1}^{n+1}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j)$$

Więc na mocy zasady o indukcji teza jest spełniona. □

3 Algorytm

Należy podzielić zadania ze względu na znak $a_i - b_i$ ($\Theta(n)$). Zadania ze znakiem dodatnim należy posortować względem b_i, a pozsotałe posortować względem a_i ($\Theta(n \log(n))$). Następnie należy wybrać zadania z drugiej puli wraz z rosnącym a_i, a później zadania z dodatnim znakiem wraz z malejącym b_i ($\Theta(n)$). Więc algorytm działa w czasie $\Theta(n \log(n))$.

4 Uzasadnienie

Weźmy optymalne rozwiązanie (a i b) i spróbujmy sprowadzić je do naszego rozwiązania (a' i b'). Weźmy pierwsze zadanie z naszego rozwiązania - a'_i, które różni się miejscami (kolejnością wykonywania) z rozwiązaniem optymalnym. Skoro jest to pierwsze czyli prefiks kolejności zadań jest taki sam do tego miejsca, więc pozycja tego zadania w rozwiązaniu optymalnym jest większa - a_k, gdzie $k > i > 0$. Więc zamieńmy miejscami a_k z a_{k-1} i sprawdźmy jak to wpłynie na nasz wynik - $max_{i=1}^{n}(\sum_{j=1}^{i} a_j - \sum_{j=1}^{i-1} b_j)$, interesuje nas jedynie zmiana sumy dla $i = k - 1$ oraz $i = k - 1$, ponieważ na poprzednie i następne i nie ma to wpływu (nie ma tych wyrazów lub są one spermutowane tylko).

Chcę pokazać, że takie przestawienie nie pogarsza wyniku czyli po zredukowaniu wyrazów, które są w obu sumach otrzymujemy:

$$max(a_k, a_{k-1} + a_k - b_k) \leqslant max(a_{k-1}, a_{k-1} + a_k - b_{k-1})$$

(Wynik ten nie może być mniejszy bo zabraliśmy optymalny rozwiązanie, ale znak ten ułatwia rachunki).

Rozważmy przypadki:
1. $a_k - b_k \leqslant 0$

a) $a_{k-1} - b_{k-1} \leqslant 0$
Więc wtedy $a_k \leqslant a_{k-1}$, ponieważ wpp. a_{k-1} byłoby wcześniej w naszym rozwiązaniu. A także $a_{k-1} + a_k - b_k \leqslant a_{k-1}$, ponieważ $a_k - b_k \leqslant 0$, więc zachodzi nierówność.

b) $a_{k-1} - b_{k-1} > 0$ Więc wtedy $a_k < a_{k-1} + a_k - b_{k-1}$, ponieważ $a_{k-1} - b_{k-1} > 0$, a także $a_{k-1} + a_k - b_k \leqslant a_{k-1}$, ponieważ $a_k - b_k \leqslant 0$, więc zachodzi nierówność.

2. $a_k - b_k > 0$
Wtedy $a_{k-1} - b_{k-1} > 0$ oraz $b_k \geqslant b_{k-1}$, ponieważ wpp. a_{k-1} byłoby wcześniej w naszym rozwiązaniu. $a_k \leqslant a_{k-1} + a_k - b_{k-1}$, ponieważ $a_{k-1} - b_{k-1} > 0$, a także $a_{k-1} + a_k - b_k \leqslant a_{k-1} + a_k - b_{k-1}$, ponieważ $b_k \geqslant b_{k-1}$

Mamy skończony ciąg więc za pomocą skonczeniu wielu takich operacji możemy otrzymać nasze rozwiązanie nie pogarszając wyniku, więc nasze rozwiązanie jest też rozwiązaniem optymalnym.