Rozwiązanie zadania 2.7

Łukasz Klasiński

18 kwietnia 2018

Spis treści

1. Problem

System złożony z dwóch maszyn A i B wykonuje n zadań. Każde z zadań wykonywane jest na obydwu maszynach, przy czym wykonanie zadania na maszynie B można rozpocząć dopiero po zakończeniu wykonywania go na maszynie A. Dla każdego zadania określone są dwie liczby naturalne a_i i b_i określające czas wykonania i-tego zadania na maszynie A oraz B (odpowiednio). Ułóż algorytm ustawiający zadania w kolejności minimalizującej czas zakończenia wykonania ostatniego zadania praz maszynę B.

2. Algorytm

2.1 Idea

W rozwiązaniu zadania skorzystamy z poniższych lematów i obserwacji.

Lemat 1.

Każde rozwiązanie optymalne da się przekształcić tak, aby kolejność wykonywania zadań na maszynie A i B była taka sama.

Lemat 2.

Niech

$$S_1 = \{a_1, \dots, a_n, b_1, \dots, b_n\}$$
$$S_2 = \{a_1 - d, \dots, a_n - d, b_1 - d, \dots, b_n - d\}$$

gdzie $d \leqslant min(S1)$ oraz R_1, R_2 będą optymalnymi rozwiązaniami dla kolejno S_1 i S_2. Wtedy użycie uporządkowania $R_1|R_2$ na zbiorze $S_2|S_1$ nie pogorszy czasu optymalnego rozwiązania.

Lemat 3a.

Jeśli czas trwania i-tego zadania $a_i = 0, a_i \in A$, to można przenieść je wraz z odpowiadającym mu zadaniem wykonywanym na maszynie B na początek kolejki, bez pogorszenia całkowitego czasu wykonywania zadań.

Lemat 3b.

Jeśli czas trwania i-tego zadania $b_i = 0, b_i \in B$, to można przenieść je wraz z odpowiadającym my zadaniem wykonywanym na maszynie A na koniec kolejki, bez pogorszenia całkowitego czasu wykonywania zadań.

Obserwacja 1. Zauważmy, że z lematu 2 oraz 3a i 3b, natychmiast wnioskujemy, że w rozwiązaniu optymalnym przestawienie najmniejszego elementu, zależnie czy jest to zadanie maszyny A|B kolejno na początek|koniec rozwiązania, zachowuje optymalny czas wykonania zadań.

Obserwacja 2. Załóżmy, że w zbiorze S mamy pary zadań a_i, b_i, a zbiór O jest pusty. Wyjmijmy ze zbioru S parę z największym elementem. Dodajmy ją do zbioru O i usuńmy z S. Następnie przesuńmy na początek|koniec listy, zależnie czy wybraliśmy element z pary, należący do $A|B$. Oczywiście po dodaniu pierwszej pary O wykonuje się optymalnie. Powtarzajmy, aż zbiór S będzie pusty. Tak budowany zbiór O po dodaniu kolejnych par, przesunie je na początek|koniec listy, a ponieważ będzie ona najmniejsza to z obserwacji 1 wiemy, że takie przesunięcie zachowuje optymalny czas wykonania zadań. Zatem po opróżnieniu zbioru S, zbiór O będzie optymalny.

2.2 Pseudokod

Zauważmy, że zbiór O z obserwacji 2, jest do pewnego momentu posortowany rosnąco po elemencie $a_i \in A$ z pary (a_i, b_i), a potem malejąco względem elementu b_i. Zatem możemy skonstruować algorytm, który będzie porządkować te pary zależnie od tego który element był mniejszy w dwóch oddzielnych zbiorach L i R, które po wykorzystaniu wszystkich zadań będą połączone i utworzą O.

```
Data: S; S = {a_1, ..., a_n, b_1, ..., b_n}
Result: Lista zawierająca kolejne indeksy zadań do wykonania
        na maszynie A i B
L, R = [];
while S niepuste do
    m := min(S) ;
    i := index(m);
    if m ∈ A then
        L.push_back(i);
    end
    else
        R.push_front(i);
    end
    S = S \ a_i;
    S = S \ b_i;
end
return concat(L, R)
```

3. Dowód

D-d lematu 1.

Weźmy dowolne rozwiązanie optymalne O i należący do niego element a_i taki, że $a_i \in A$. Niech $b_k \in B$ będzie pierwszym elementem, który rozpoczyna się po a_i. Wszystkie zadania od b_k włącznie do b_i nazwijmy sekcją. Wyjmijmy z rozwiązania b_i. Teraz możemy przesunąć sekcję o czas wykonywania b_i, ponieważ przesunięcie ich w prawo nie zmieni czasu rozwiązania. W ten sposób utworzyła się przerwa wielkości b_i po a_i, gdzie wstawiamy usunięty wcześniej element.

□

D-d lematu 2.

Weźmy S_1,S_2 z lematu. Niech R_1 będzie optymalnym rozwiązaniem dla S_1, a R_2 dla S_2. Przejście z S_1 do S_2 nazwijmy ***kurczeniem***, a z S_2 do S_1 ***rozciąganiem***. Pokażemy, że $R_1|R_2$ jest optymalne dla $S_2|S_1$.

Obserwacja ***rozciąganie, kurczenie***

Podczas *rozciągania* powiększamy elementy z A oraz B o jakąś stałą $d \leqslant min(S_1)$. Ponieważ powiększamy n elementów z A i B, a ostatni element z B określa, kiedy zakończy się czas wykonania zadań, to po rozszerzeniu ostatniego elementu o d w najgorszym przypadku czas wydłuży się o co najwyżej $d * (n+1)$.

Podobnie w przypadku *kurczenia* czas skróci się o co najwyżej $d * (n+1)$.

Załóżmy nie wprost, że $\exists R_1$, że

$$T(S_2, R_2) < T(S_2, R_1)$$

oraz

$$T(S_1, R_1) < T(S_2, R_1)$$

gdzie $T(S, R)$ oznacza czas wykonania zadań S w kolejności R.

Wtedy z *rozciągania* mamy:

$$T(S_2, R_2) + (n+1)d \geqslant T(S_1, R_2)$$

natomiast z *kurczenia*:

$$T(S_1, R_1) - (n+1)d \geqslant T(S_2, R_1)$$

Po kilku prostych przekształceniach dochodzimy do sprzeczności:

$$\begin{gathered}
T(S_2, R_1) \leqslant T(S_1, R_1) - (n+1)d \\
T(S_2, R_2) \geqslant T(S_1, R_2) + (n+1)d \\
T(S_1, R_1) - (n+1)d \geqslant T(S_2, R_2) > T(S_2, R_2) \geqslant T(S1, R2) - (n+1)d \\
T(S_1, R_2) - (n+1)d > T(S_1, R_2) - (n+1)d \\
T(S_1, R_2) > T(S_1, R_2) \\
\perp
\end{gathered}$$

D-d lematu 3a. Zadanie a_i możemy przestawić na początek kolejki maszyny A, ponieważ nic nie trwa, zatem nie przesunie pozostałych zadań. Natomiast odpowiadające zadanie b_i podobnie jak w dowodzie lematu 1 możemy wyciągnąć, przesunąć poprzedzającą go sekcję o jego czas trwania i w wstawić na początek kolejki maszyny B bez pogorszenia całkowitego czasu działania. $\square$

D-d lematu 3b. Podobnie jak w lemacie 3a, zadanie b_i możemy przestawić na koniec, ponieważ nic nie trwa, więc nie wydłuży czasu wykonania, natomiast odpowiadające mu zadanie a_i możemy wyciągnąć, następujące po nim zadania przesunąć w lewo o czas jego trwania i wstawić na koniec kolejki A. $\square$