Systemy wbudowane - wykład do samodzielnego opracowania
Systemy czasu rzeczywistego (2)

Przemek Błaśkiewicz

6 maja 2020

Definicja

System czasu rzeczywistego

Real-time system to system, którego poprawność działania zależy nie tylko od poprawności rezultatów (obliczeń, decyzji, akcji), ale również od czasu, kiedy zostaną one wypracowane (*czas reakcji*).

Systemy RT charakteryzują się *gotowością* i *responsywnością*. To pojęcie nie oznacza „szybki", ale „pewny".
Działanie w obostrzeniach czasowych możliwe jest, ponieważ projektant systemu ma (powinien mieć) szczegółową i pełną wiedzę o możliwych zdarzeniach, co pozwala na uszeregowanie (zaplanowanie) zadań wykonywanych w systemie pod względem czasu trwania, priorytetu, okresowości...

Plan

Plan (*schedule*) dla zbioru zadań $(J_1, J_2, \ldots, J_n)$ to pewna funkcja: $\sigma : R^+ \to \{0, \ldots, n\}$, taka, że:

$$\forall t \in R^+, \exists t_1, t_2 \in R^+ : t \in [t_1, t_2), \forall t' \in [t_1, t_2) : \sigma(t) = \sigma(t').$$

Realizowalność planu

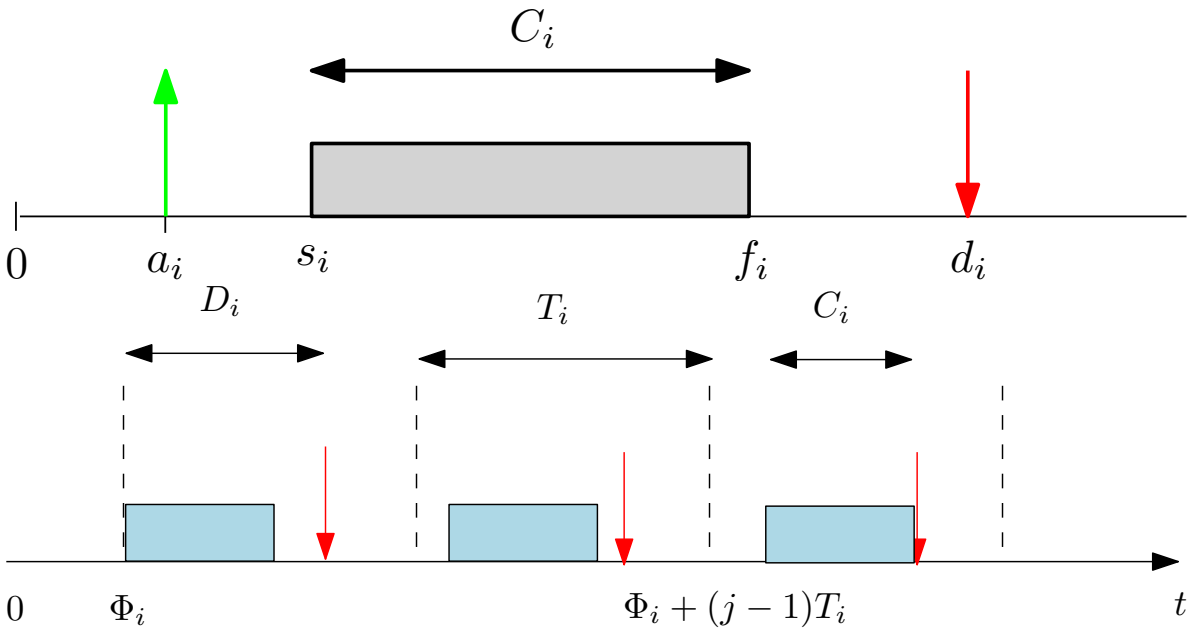
(*feasibility*) – plan jest realizowalny, jeśli jego wykonanie powoduje, że wszystkie zadania kończą się nie później niż w swoim terminie wykonania.

Lemat o wywłaszczeniu

Jeśli czasy nadejścia zadań są synchroniczne, to mechanizm wywłaszczenia nie polepsza maksymalnego opóźnienia. Czyli jeśli istnieje algorytm wywłaszczeniowy o opóźnieniu L_{max} to istnieje również algorytm niewywłaszczeniowy o tym samym opóźnieniu dla tego samego zestawu zadań.

Cechy charakterystyczne zadań

Zadania w systemie mogą mieć różne właściwości dotyczące momentów ich pojawiania się. W systemach monitorujących są to zazwyczaj zadania okresowe ♠ *polling*, oraz pojawiające się nagle (w formie przerwań).



a_i – nadejście zadania, d_i – deadline, C_i – czas wykonywania, s_i – początek wykonywania, f_i – koniec wykonywania. ($d_i - f_i$ – ♠??), D_i – przedział czasu do wykonania, T_i – okres, Φ_i – faza (iteracja).

Algorytmy planowania

Systemy mogą realizować różne polityki wykonywania zadań:

- wywłaszczające i niewywłaszczające
  albo podział czasu: każde zadanie może wykonywać się przez określoną jednostkę czasu, a następnie jest wymieniane, albo zadanie wykonywane jest tak długo, jak w systemie nie ma innego gotowego do wykonania o wyższym priorytecie
- statyczne i dynamiczne;
  plan jest zapisany na stałe, lub tworzony ad-hoc na podstawie aktualnej sytuacji systemu
- jednoprocesorowe, wieloprocesorowe;
- optymalne, heurystyczne;
  niektóre przypadki są trudne (algorytmicznie) do zaplanowania...
- dla zadań okresowych (periodycznych) i asynchronicznych;
  w systemach może nie być zadań periodycznych, albo zadań nagłych...

Zadania aperiodyczne, synchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - dla wszystkich zadań $a_i = 0, i = 1, \ldots n$; (synchroniczność)
  - każde zadanie ma określony deadline d_i i czas obliczenia C_i;
  - zadania są niezależne (nie ma zależności wykonania).
- system bez wywłaszczenia
- jednoprocesorowy
- zadanie: znajdź optymalny plan

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

To planista „studencki" – robi projekty w takiej kolejności, w jakiej trzeba je oddawać.

♠ Zastanów się jednak: robiąc zadania „po studencku" nie realizujesz pełnego EDD – przesuwasz s_i maksymalnie do przodu tak, aby $f_i = d_i$. Co to daje?

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 1

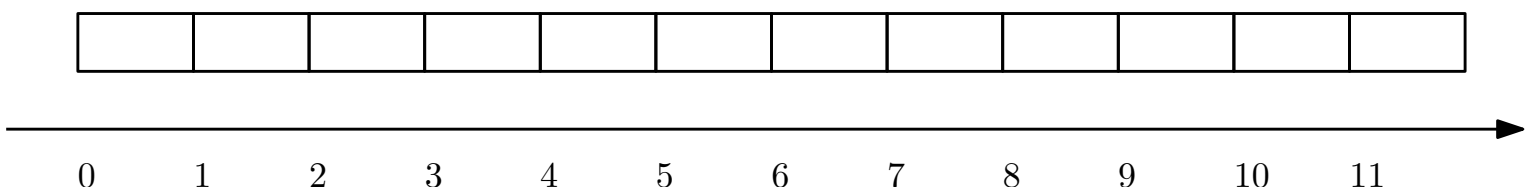
	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 1

	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5



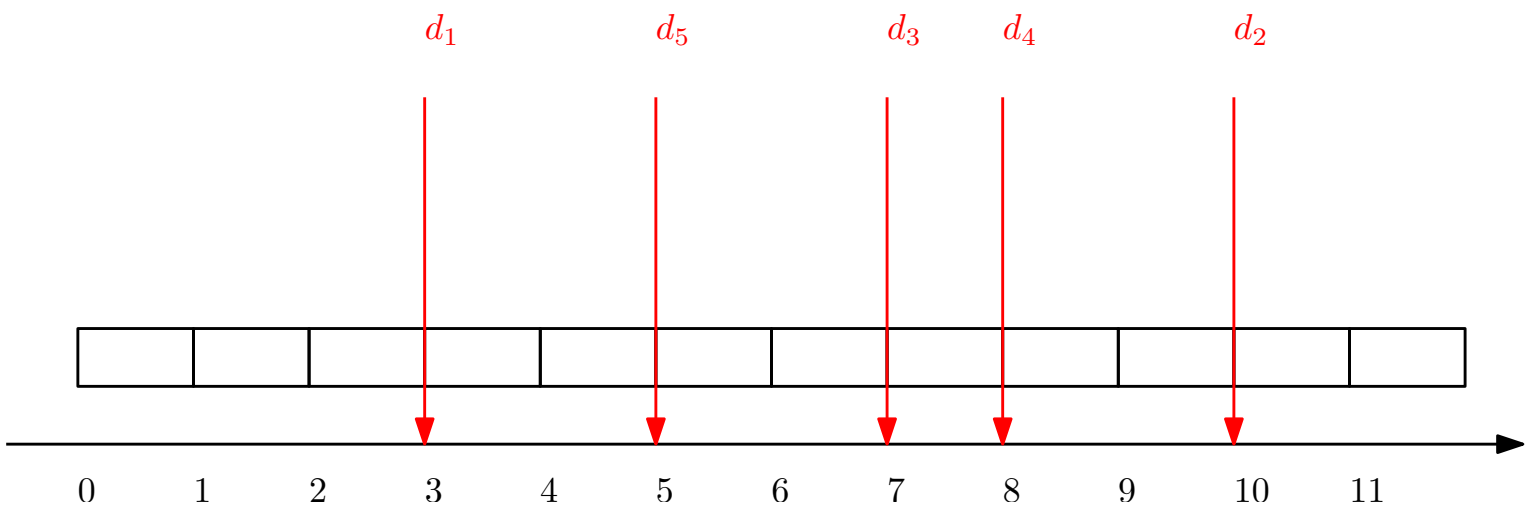
Ramy czasowe

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 1

	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5



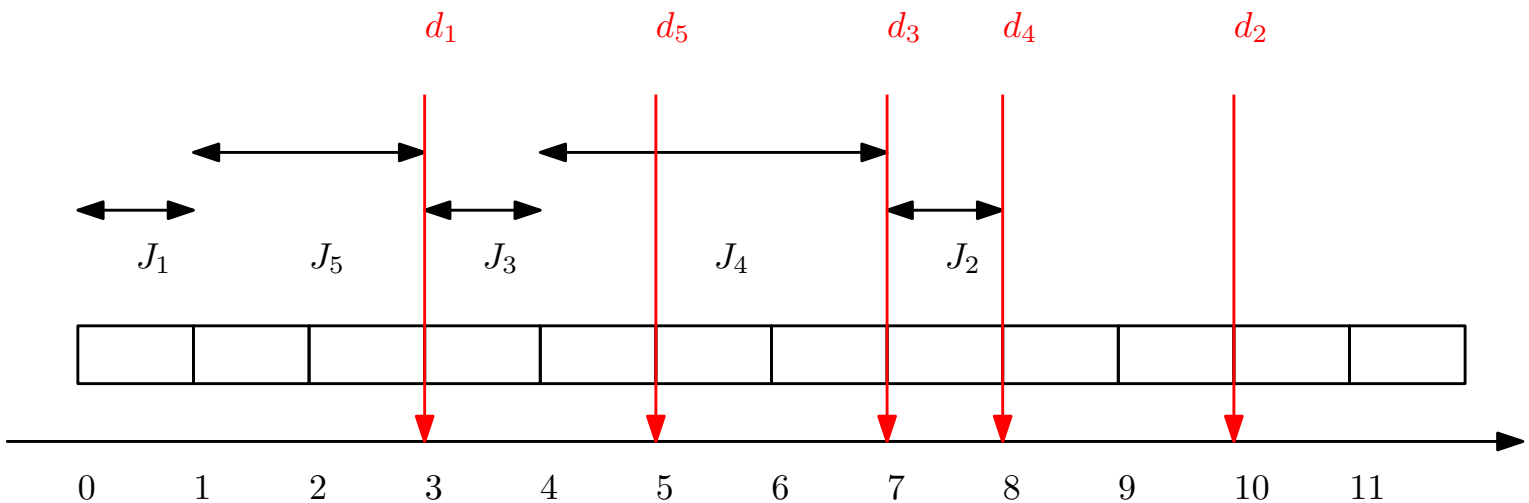
Punkty krytyczne

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 1

	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5



Sortujemy: $(J_1, J_5, J_3, J_4, J_2)$ i upychamy od lewej...

Żadne zadanie nie przekracza swojego deadline!

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 2

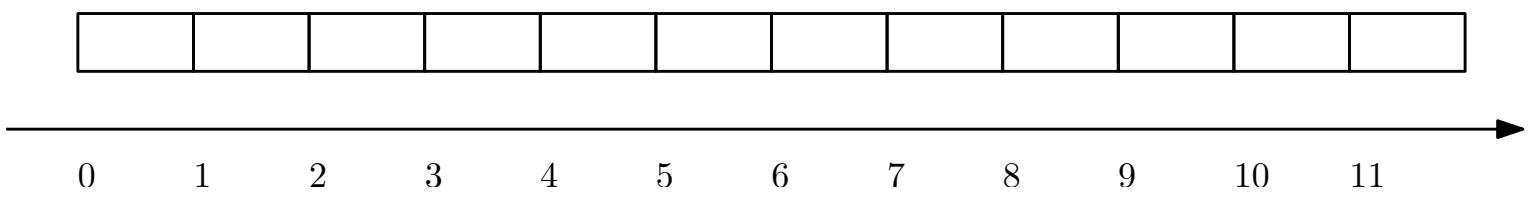
	J_1	J_2	J_3	J_4	J_5
C_i	1	**2**	1	**4**	2
d_i	**2**	**5**	**4**	8	**6**

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 2

	J_1	J_2	J_3	J_4	J_5
C_i	1	**2**	1	**4**	2
d_i	**2**	**5**	**4**	8	**6**



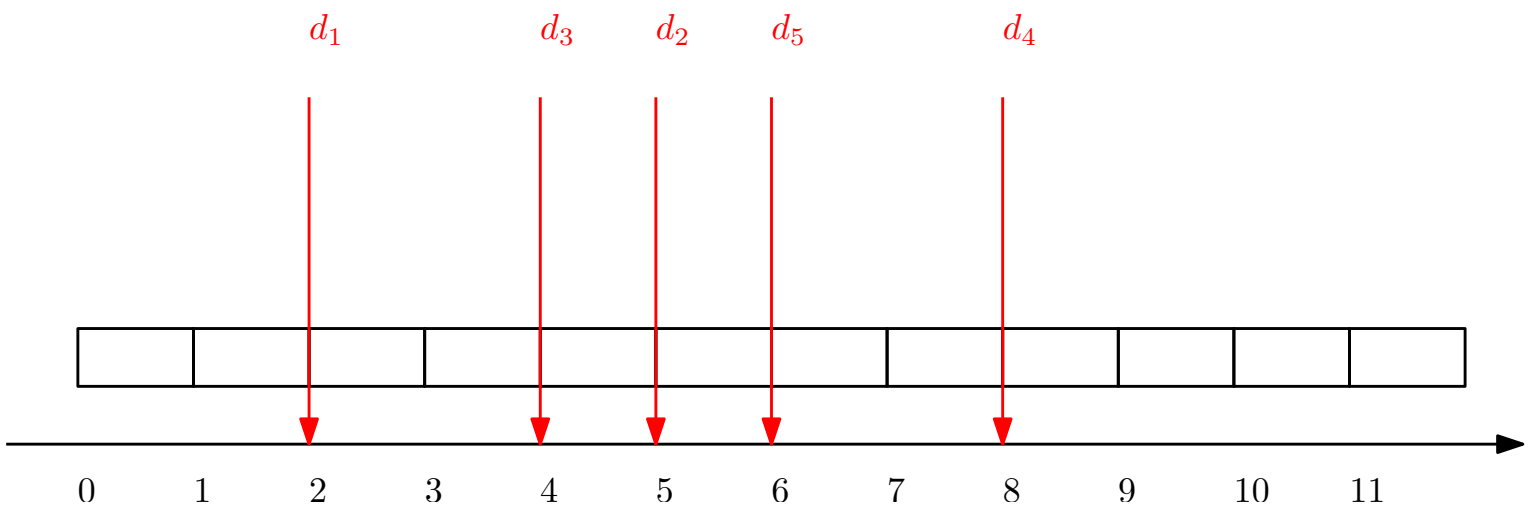
Ramy czasowe

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 2

	J_1	J_2	J_3	J_4	J_5
C_i	1	**2**	1	**4**	2
d_i	**2**	**5**	**4**	8	**6**



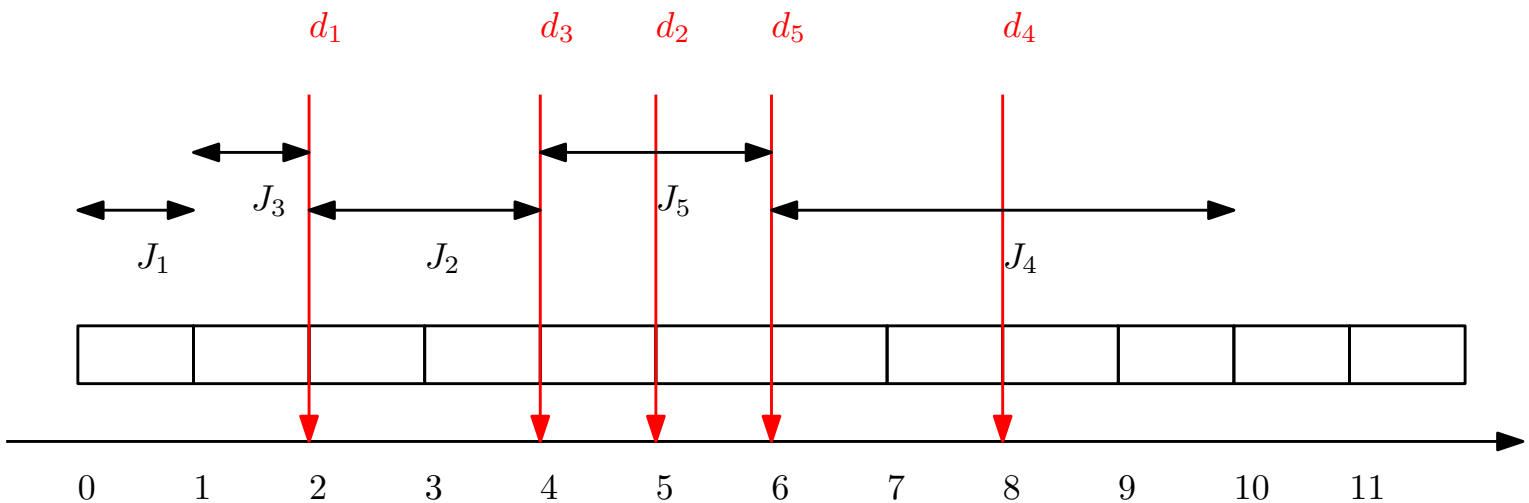
Punkty krytyczne

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Przykład 2

	J_1	J_2	J_3	J_4	J_5
C_i	1	**2**	1	**4**	2
d_i	**2**	**5**	**4**	8	**6**



Sortujemy: $(J_1, J_3, J_2, J_5, J_4)$ i upychamy od lewej...

Zadanie J_4 przekracza deadline i to o 2 jednostki!

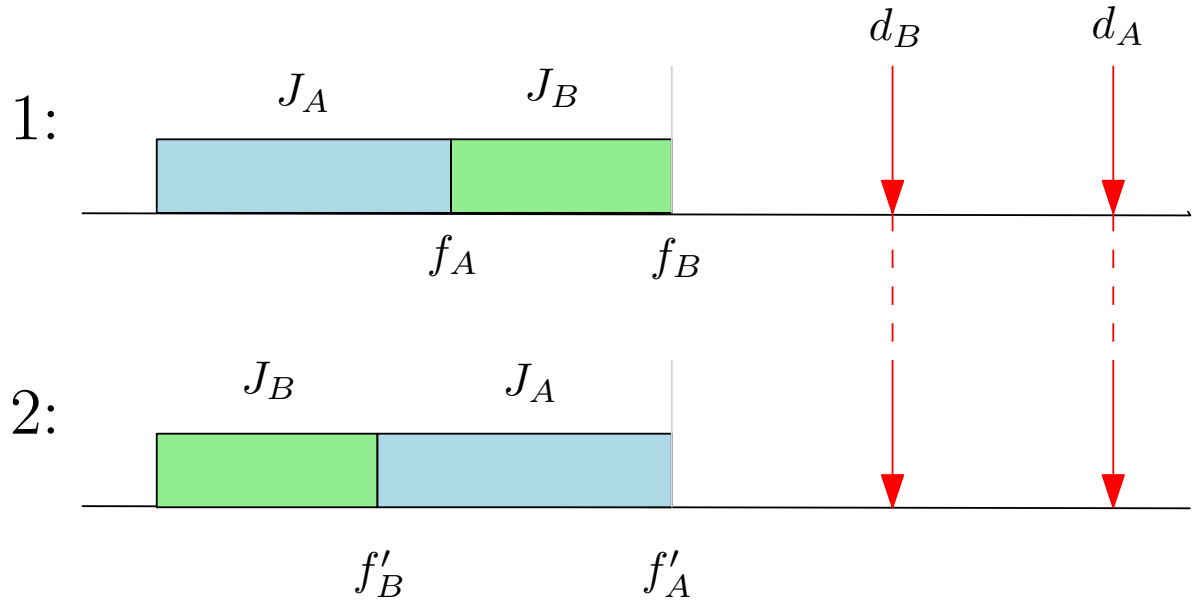
Twierdzenie o optymalności EDD

Zadań z przykładu 2. nie można włożyć w realizowalny plan.
♠ Co można zmienić, by zestaw był możliwy do zaplanowania? Jakie to niesie konsekwencje dla całego systemu?
Mimo tego kolejność zadań: 1, 3, 2, 5, 4 jest *optymalna* (nie da się ich uszeregować lepiej, tj. tak, by opóźnić je o mniej niż 2 jednostki).

Jackson, 1955 – Optymalność EDD

Jeśli dany jest dowolny zestaw niezależnych zadań, pojawiających się w sytemie synchronicznie, to każdy algorytm wykonujący je w kolejności niemalejących terminów wykonania (deadlines) jest algorytmem optymalnym ze względu na minimalizację maksymalnego opóźnienia.

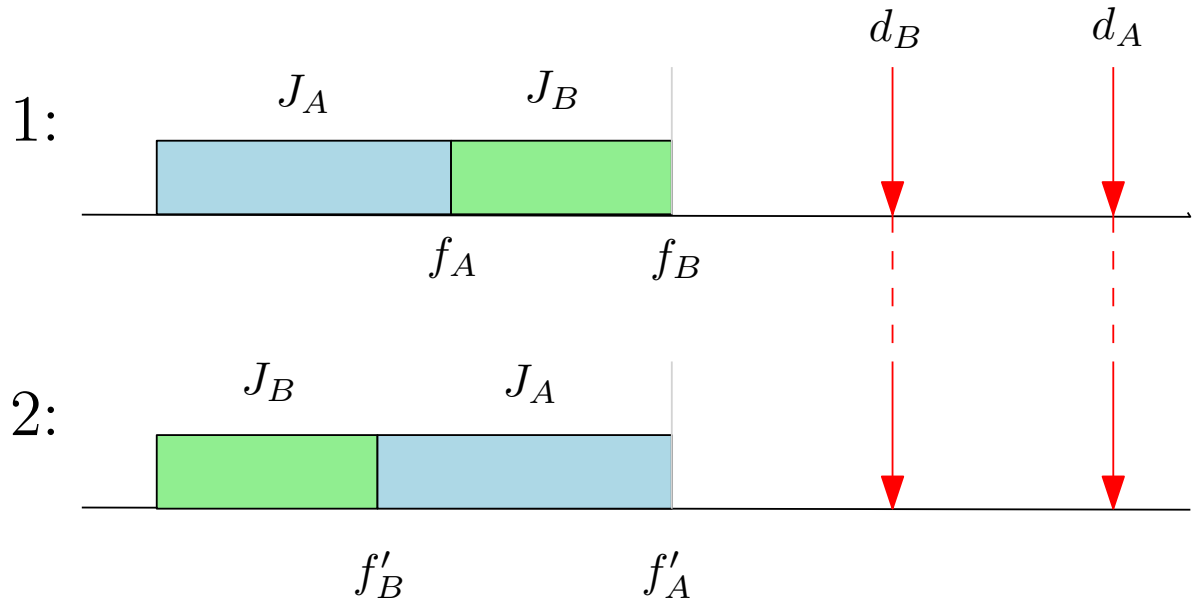
Twierdzenie o optymalności EDD – dowód ♠



Dla dwóch zadań ustalmy dwa porządki: (1) niezgodny z EDD i zgodny (2). Zauważmy, że $f_B = f'_A$: koniec pracy jest niezależny od kolejności wykonywania zadań. Przypomnijmy, *lateness* to różnica między końcem wykonania zadania a jego *deadline* – dodatnie lateness oznacza opóźnienie, im bardziej ujemne, tym zadanie ma więcej „luzu”:

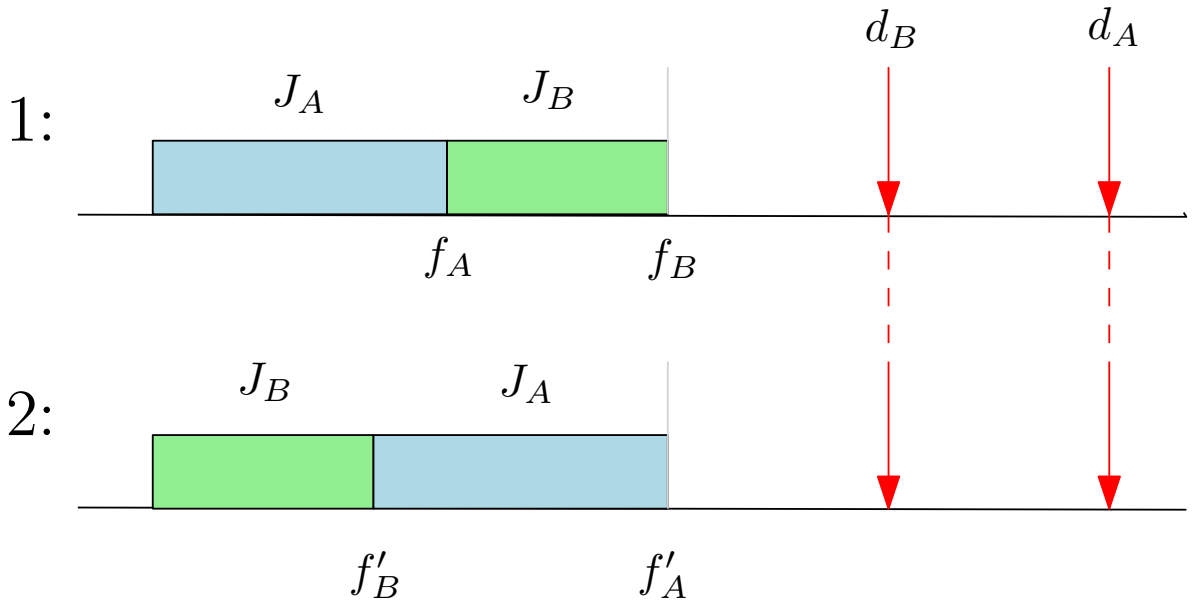
$$L_A = f_A - d_A.$$

Twierdzenie o optymalności EDD – dowód ♠



- (A) $f'_A - d_A = f_B - d_A \leq f_B - d_B$ (L'_A niewiększe niż L_B)

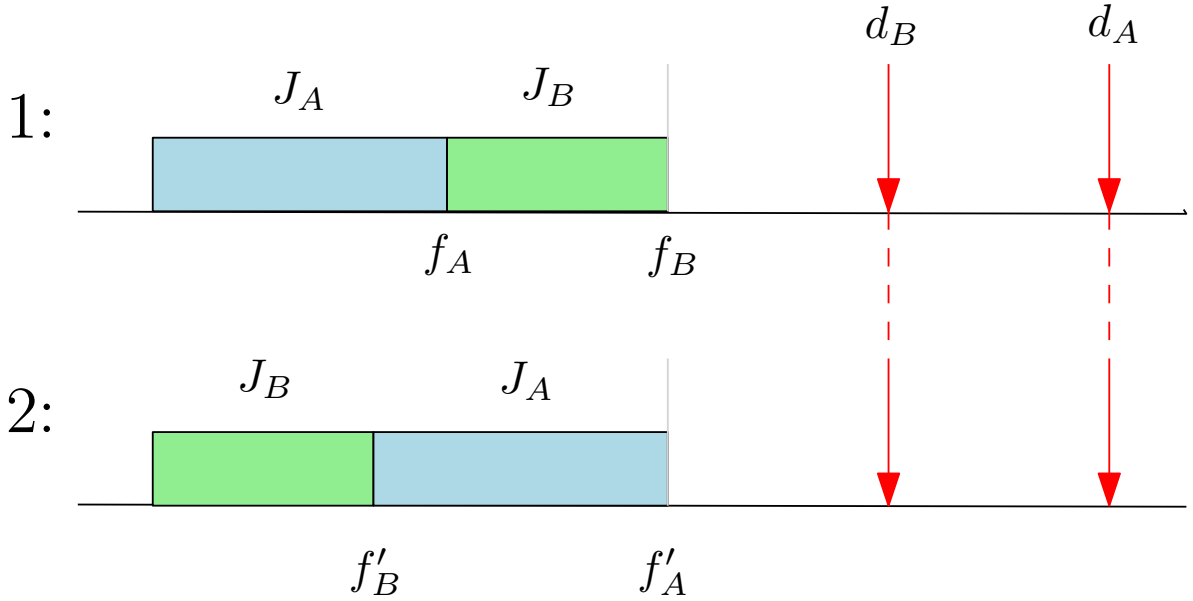
Twierdzenie o optymalności EDD – dowód ♠



A $f'_A - d_A = f_B - d_A \leq f_B - d_B$ (L'_A niewiększe niż L_B)

B $f'_B - d_B \leq f_B - d_B$ (L'_B niewiększe niż L_B)

Twierdzenie o optymalności EDD – dowód ♠

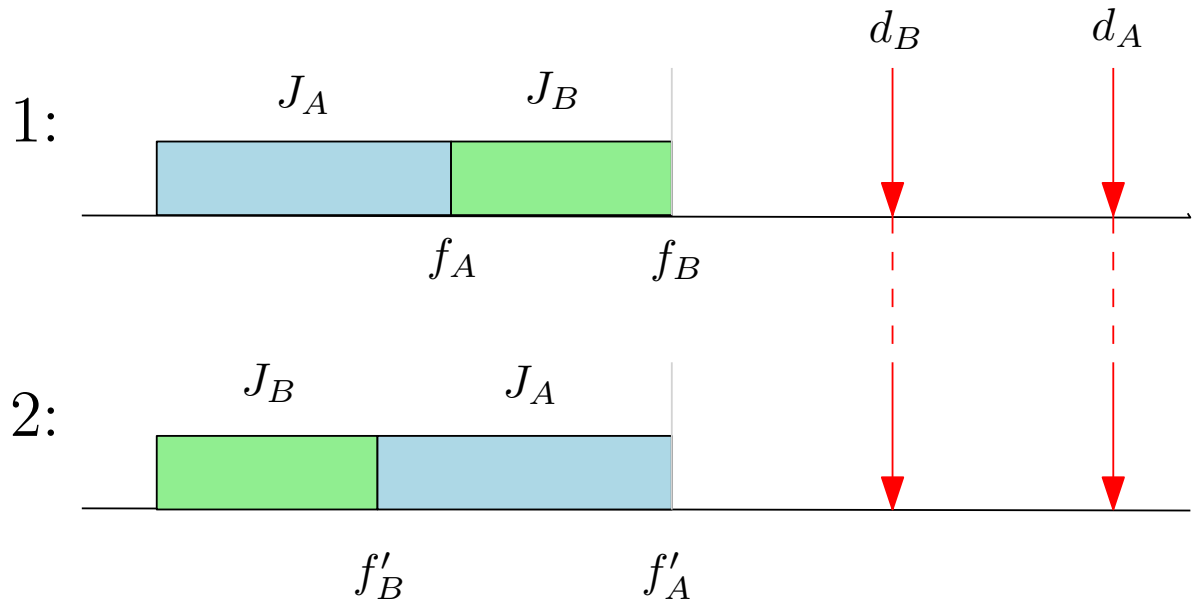


A. $f'_A - d_A = f_B - d_A \leq f_B - d_B$ (L'_A niewiększe niż L_B)

B. $f'_B - d_B \leq f_B - d_B$ (L'_B niewiększe niż L_B)

C. Ponieważ w (1) $L_B \leq L_A$, to L_B jest lepszym ograniczeniem

Twierdzenie o optymalności EDD – dowód ♠



A. $f'_A - d_A = f_B - d_A \leq f_B - d_B$ (L'_A niewiększe niż L_B)

B. $f'_B - d_B \leq f_B - d_B$ (L'_B niewiększe niż L_B)

C. Ponieważ w (1) $L_B \leq L_A$, to L_B jest lepszym ograniczeniem

zatem: $L'_{max} \leq L_{max}$ (indeksy a,b)

EDD - właściwości

- złożoność EDD: sortowanie, $\mathcal{O}(n \log n)$
- test na *planowalność* zadań $\{J_1, \ldots, J_n\}$:
  1. posortuj listę zadań niemalejąco względem ich deadlines; (niech to będzie powyższy porządek);
  2. sprawdź, czy $\forall i \in 1, \ldots, n : \; f_i \leq d_i$;
     - ponieważ $f_i = \sum_{k=1}^{i} C_k$ warunek wystarczający:

$$\forall i \in 1, \ldots, n : \sum_{k=1}^{i} C_k \leq d_i$$

- EDD jest optymalny, zatem jeśli zestawu zadań nie można rozplanować przez EDD, to nie można go rozplanować w ogólności.

Zadania aperiodyczne, asynchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - a_i dla wszystkich zadań jest różne i nieznane;
  - każde zadanie ma określony deadline d_i i czas obliczenia C_i;
  - zadania są niezależne (nie ma zależności wykonania).
- system z wywłaszczeniem;
- jednoprocesorowy
- zadanie: znajdź plan minimalizujący maksymalne opóźnienie.

Planista EDF – Earliest Deadline First

EDF

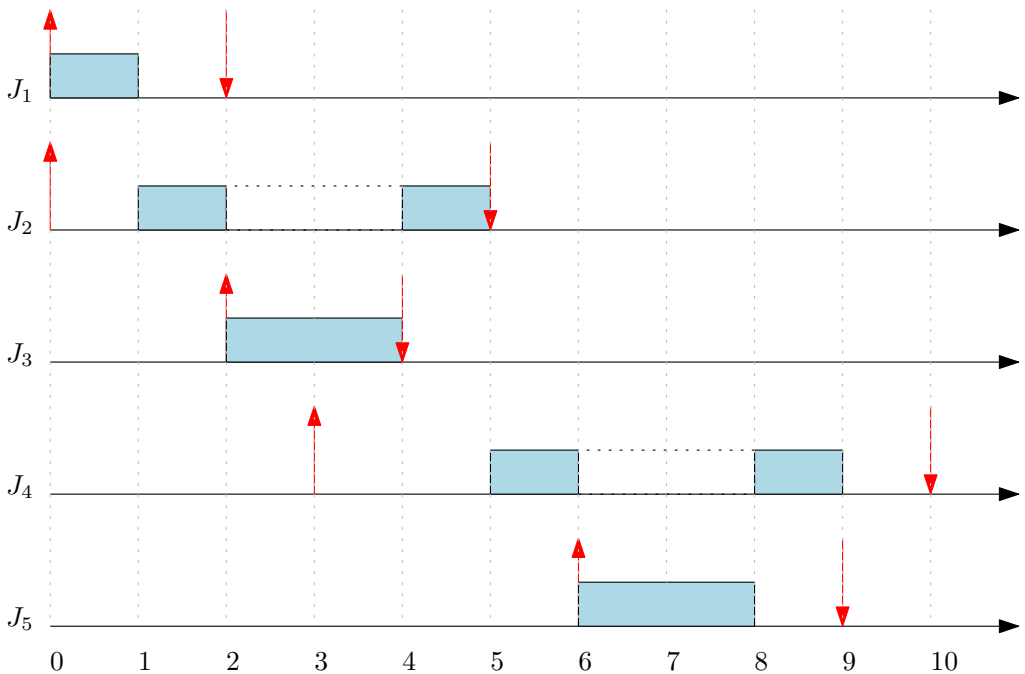
W każdym momencie wykonuj zadanie z najwcześniejszym czasem wykonania spośród dostępnych zadań w systemie.

- Jeśli w systemie pojawia się zadanie z czasem wykonania wcześniejszym niż aktualnie wykonywane zadanie - wykonywane zadanie jest wywłaszczone na rzecz nowo przyszłego zadania.
- Założenie o znikomości czasu na przełączanie zadań!

To wersja „algorytmu studenta" (może bardziej realistyczna?), gdzie robicie zadania, które są na „za chwilę", ale jeśli jest jakaś wrzutka, to rzucacie wszystko i przełączacie się na nią.
♠ Jakie konsekwencje niesie to za sobą (np. w kontekście rozkładania odpowiednich podręczników na biurku...)?

EDF - przykład

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
C_i	1	2	2	2	2
d_i	2	5	4	10	9



W chwili 2 pojawia się J_3 z $d_3 = 4 < d_2$, następuje wywłaszczenie.

W chwili 4 do J_3 jest zakończone, w systemie są J_2 (część) i J_4. Ale $d_4 > d_2$, zatem wykonywane jest na powrót J_2.

W chwili 6 pojawia się J_5 i wywłaszcza J_4.

Optymalność EDF

Horn 1974

Jeśli dany jest dowolny zestaw niezależnych zadań, pojawiających się w sytemie asynchronicznie, to każdy algorytm wykonujący je tak, że w każdym momencie wykonywane jest zadanie o najbliższym czasie wykonania (deadline) jest algorytmem optymalnym ze względu na minimalizację maksymalnego opóźnienia.

– bez dowodu (mechanizm zbliżony do dowodu tw. Jacksona) –

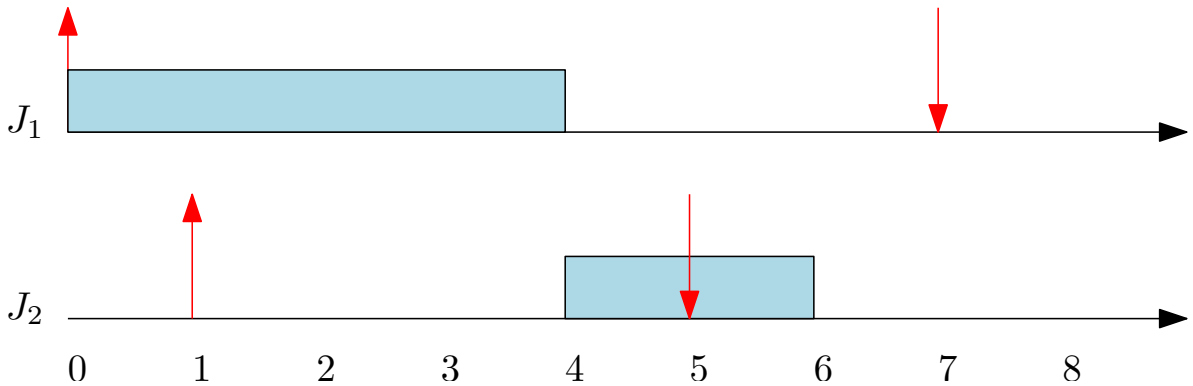
Zadania aperiodyczne, asynchroniczne, wersja bez wywłaszczenia

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - a_i dla wszystkich zadań jest różne i nieznane;
  - każde zadanie ma określony deadline d_i i czas obliczenia C_i;
  - zadania są niezależne (nie ma zależności wykonania).
- system bez wywłaszczeń;
- jednoprocesorowy
- zadanie: znajdź plan minimalizujący maksymalne opóźnienie.

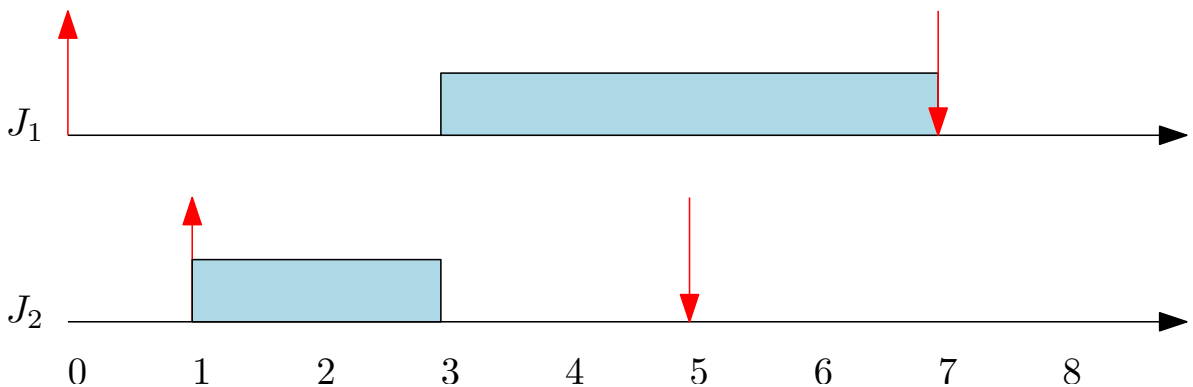
EDF bez wywłaszczeń - przykład

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

- EDF bez wywłaszczeń nie daje realizowalnego planu:



- Bo optymalnie można zrealizować oba zadania:



EDF bez wywłaszczeń - własności

- W poprzednim przykładzie CPU jest bezczynne w okresie 0-1, mimo gotowości zadania J_1.
- Jeśli czasy a_i nie są znane, to niemożliwe jest podjęcie decyzji o tej bezczynności.

Jeffay i in. 1991

Przy braku zezwolenia na okresy bezczynności przy gotowym do wykonania zadaniu, EDF jest optymalny również dla modelu bez wywłaszczeń.

Wróżenie z fusów nie jest możliwe, ale jeśli możemy rozluźnić założenia, to można już coś zdziałać...

Zadania aperiodyczne, asynchroniczne, wersja bez wywłaszczenia, dozwolone czasy bezczynności

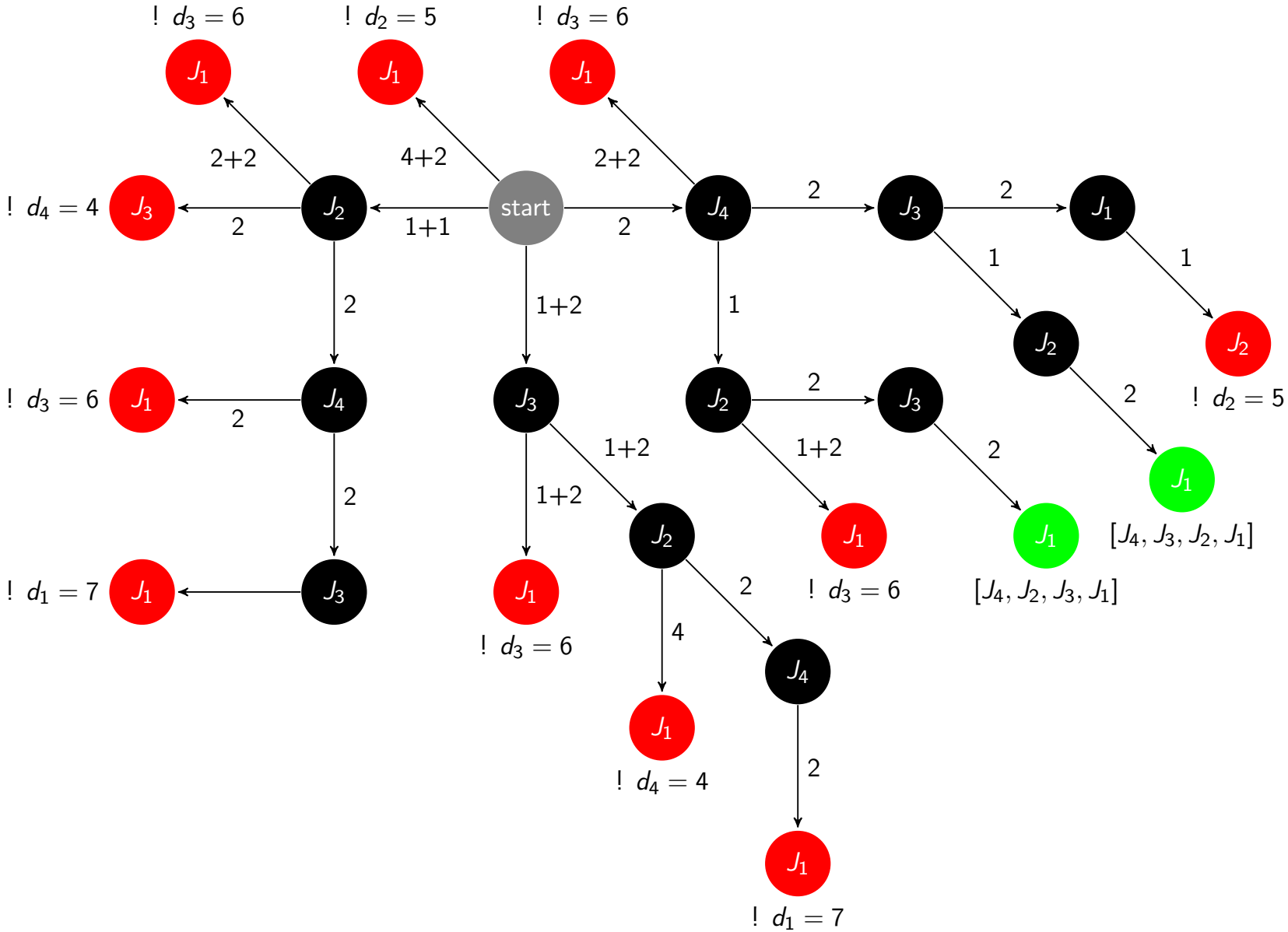
- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - a_i dla wszystkich zadań znane;
  - każde zadanie ma określony deadline d_i i czas obliczenia C_i;
  - zadania są niezależne (nie ma zależności wykonania).
- system bez wywłaszczeń: zadania są wykonywane do końca;
- system zezwala na czasy bezczynności przy gotowym zadaniu;
- jednoprocesorowy
- zadanie: znajdź plan minimalizujący maksymalne opóźnienie.

W ogólności, problem jest **NP-trudny**. Rozwiązywalny przy użyciu heurystyk lub mechanizmów typu *branch-and-bound*.

Algorytm Bratley'a

- Wykorzystując mechanizm branch-and-bound odnajduje realizowalne rozwiązanie, jeśli istnieje.
- Bazuje na odpowiedniej permutacji zestawu zadań $J_1, \ldots, J_n$.
- Rozpoczyna od pustej listy (kolejności) zadań;
- **Branch:** wybierz do listy kolejne zadanie (z pozostałych)
- **Bound:**
  1. znaleziono realizowalny plan $\rightarrow$ bieżąca lista realizowalnym rozwiązaniem;
  2. jeśli istnieje zadanie, którego *deadline* minął, a nie zostało jeszcze zaplanowane $\rightarrow$ bieżąca ścieżka to nierealizowalny plan (cofnij do ostatniego dokonanego wyboru).

	J_1	J_2	J_3	J_4
a_i	4	1	1	0
C_i	2	1	2	2
d_i	7	5	6	4



J_1: a=4, C=2, d=7; J_2: a=1, C=1, d=5;
J_3: a=1, C=2, d=6; J_4: a=0, C=2, d=4

Zadanie ♠

Przeanalizuj sposób wyszukiwania realizowalnego planu za pomocą algorytmu Bratley'a na podstawie danych z poprzedniego slajdu. Węzły oznaczają poszczególne zadania. Etykiety na krawędziach oznaczają czas, który musi upłynąć, aby można było zakończyć dane zadanie z końca krawędzi. Jeśli etykieta jest w postaci $N + M$, to oznacza to, że trzeba bezczynnie odczekać N jednostek i następnie wykonywać zadanie przez M jednostek czasu.

Algorytm Bratley'a - własności

- Złożoność wykładnicza – tylko jako algorytm offline;
- Odkryty schemat wykonania zapisany jako lista i odczytywany w kolejności przez planistę w trakcie działania systemu.
- Realizacja – dobry przykład zastosowania struktury drzewa w programowaniu :-)