Systemy wbudowane - wykład 10

Przemek Błaśkiewicz

10 maja 2018

Notes

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*).

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.

Notes

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.
- Współdziała ze środowiskiem, reagując na jego niezależne zmiany.

Notes

Przypomnienie o RTOS

- Pojęcie "real-time" nie oznacza "szybki".

Notes

Przypomnienie o RTOS

- Pojęcie "real-time" nie oznacza "szybki".
- PC vs. SW vs. RTOS

Notes

Przypomnienie o RTOS

- Pojęcie "real-time" nie oznacza "szybki".
- PC vs. SW vs. RTOS
- hard/soft/firm OS

Notes

Planowanie zadań

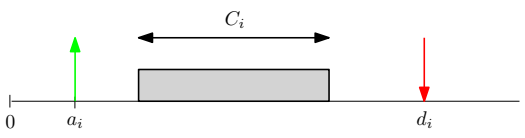
Plan

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

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

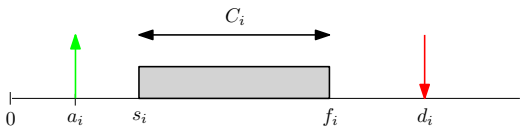
Notes

Cechy charakterystyczne zadań



Notes

Cechy charakterystyczne zadań



Notes

Algorytmy planowania (1)

Realizowalność planu

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

Notes

Algorytmy planowania (1)

Realizowalność planu

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

- wywłaszczające i niewywłaszczające;

Notes

Algorytmy planowania (1)

Realizowalność planu

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

- wywłaszczające i niewywłaszczające;
- statyczne i dynamiczne;

Notes

Algorytmy planowania (1)

Realizowalność planu

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

- wywłaszczające i niewywłaszczające;
- statyczne i dynamiczne;
- jednoprocesorowe, wieloprocesorowe;

Notes

Algorytmy planowania (1)

Realizowalność planu

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

- wywłaszczające i niewywłaszczające;
- statyczne i dynamiczne;
- jednoprocesorowe, wieloprocesorowe;
- optymalne, heurystyczne;

Notes

Algorytmy planowania (1)

Realizowalność planu

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

- wywłaszczające i niewywłaszczające;
- statyczne i dynamiczne;
- jednoprocesorowe, wieloprocesorowe;
- optymalne, heurystyczne;
- dla zadań okresowych (periodycznych) i asynchronicznych;

Notes

Lemat o wywłaszczaniu

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.

Notes

Zadania aperiodyczne, synchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:

Notes

Zadania aperiodyczne, synchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - dla wszystkich zadań $a_i = 0, i = 1, \ldots n$;

Notes

Zadania aperiodyczne, synchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - dla wszystkich zadań $a_i = 0, i = 1, \ldots n$;
  - każde zadanie ma określony deadline d_i i czas obliczenia C_i;

Notes

Zadania aperiodyczne, synchroniczne

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

Notes

Zadania aperiodyczne, synchroniczne

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

Notes

Zadania aperiodyczne, synchroniczne

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

Notes

Zadania aperiodyczne, synchroniczne

- Dany jest zestaw n zadań: $J_1, \ldots, J_n$, gdzie:
  - dla wszystkich zadań $a_i = 0, i = 1, \ldots n$;
  - 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

Notes

Planista EDD (earliest due date)

Notes

Planista EDD (earliest due date)

Wykonuje zadania w kolejności niemalejącego czasu zakończenia.

Notes

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

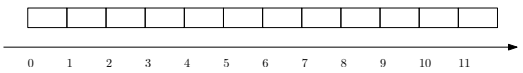
Notes

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



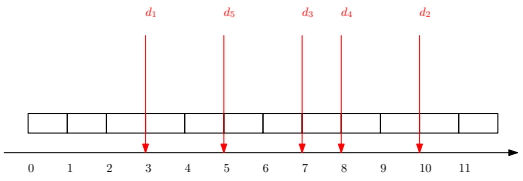
Notes

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



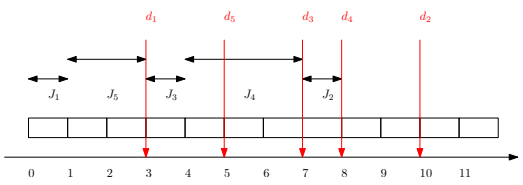
Notes

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



Notes

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

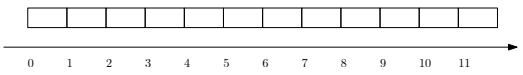
Notes

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



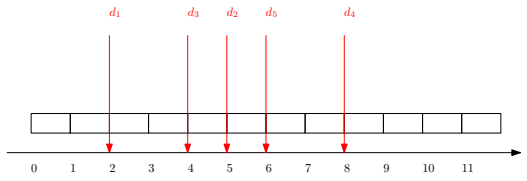
Notes

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



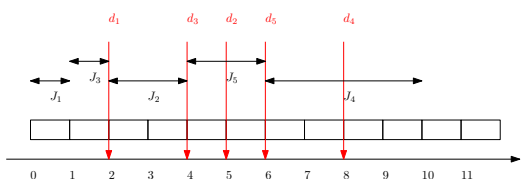
Notes

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



Notes

Twierdzenie o optymalności EDD

Notes

Twierdzenie o optymalności EDD

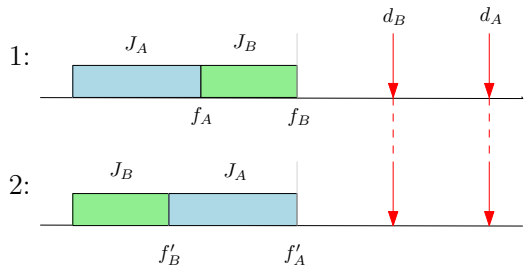
Notes

Jackson, 1955

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.

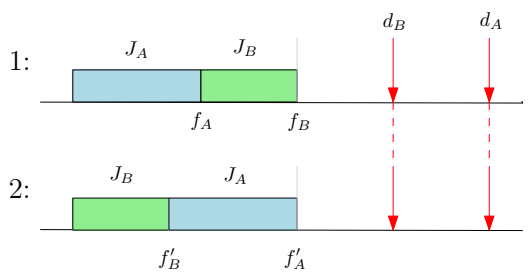
Twierdzenie o optymalności EDD

Notes



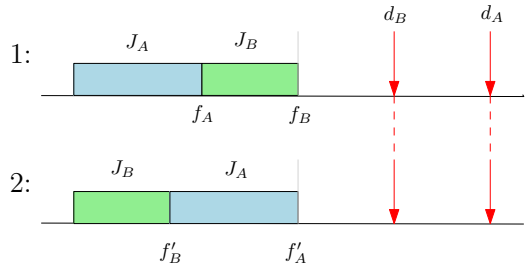
Twierdzenie o optymalności EDD

Notes



Ⓐ $f'_A - d_A = f_B - d_A \leq f_B - d_B$

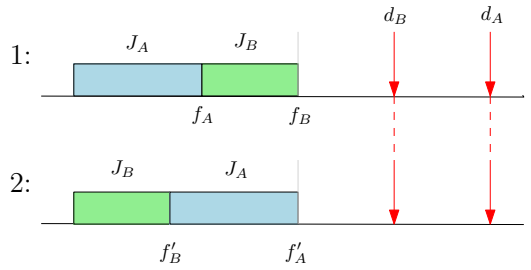
Twierdzenie o optymalności EDD



Ⓐ $f'_A - d_A = f_B - d_A \leq f_B - d_B$
Ⓑ $f'_B - d_B \leq f_B - d_B$

Notes

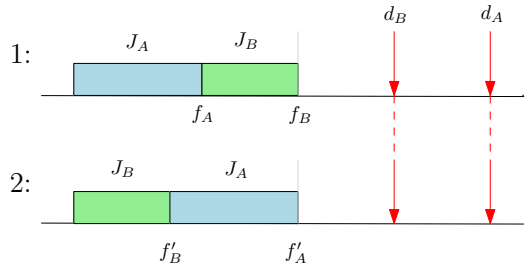
Twierdzenie o optymalności EDD



Ⓐ $f'_A - d_A = f_B - d_A \leq f_B - d_B$
Ⓑ $f'_B - d_B \leq f_B - d_B$

Notes

Twierdzenie o optymalności EDD



Ⓐ $f'_A - d_A = f_B - d_A \leq f_B - d_B$
Ⓑ $f'_B - d_B \leq f_B - d_B$

$$L'_{\substack{max \\ a,b}} \leq L_{\substack{max \\ a,b}}$$

Notes

EDD - właściwości

- złożoność EDD: sortowanie, $\mathcal{O}(n \log n)$

Notes

EDD - właściwości

- złożoność EDD: sortowanie, $\mathcal{O}(n \log n)$
- test na *planowalność* zadań $\{J_1, \ldots, J_n\}$:

Notes

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);

Notes

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 $\forall i \in 1, \ldots, n : \; f_i \leq d_i$;

Notes

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:

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

Notes

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.

Notes

Zadania aperiodyczne, asynchroniczne

Notes

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.

Notes

Planista EDF – Earliest Deadline First

EDF

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

Notes

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.

Notes

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!

Notes

EDF - przykład

Notes

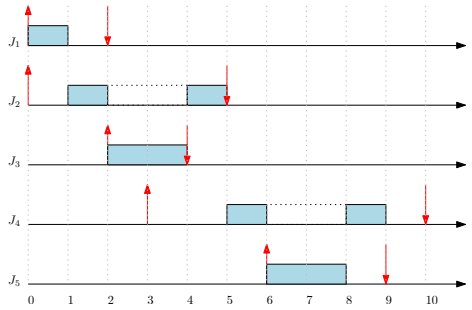
EDF - przykład

	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

Notes

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



Notes

Optymalność EDF

Notes

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.

Notes

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) –

Notes

Zadania aperiodyczne, asynchroniczne, wersja bez wywłaszczenia

Notes

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.

Notes

EDF bez wywłaszczeń - przykład

Notes

EDF bez wywłaszczeń - przykład

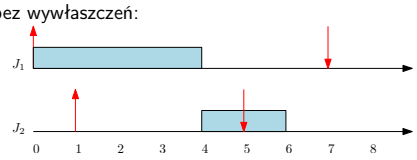
	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

Notes

EDF bez wywłaszczeń - przykład

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

- EDF bez wywłaszczeń:

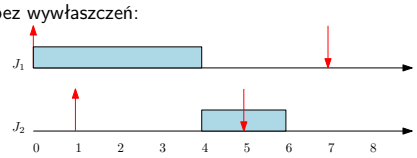


Notes

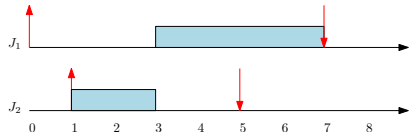
EDF bez wywłaszczeń - przykład

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

- EDF bez wywłaszczeń:



- optymalnie:



Notes

EDF bez wywłaszczeń - własności

- W poprzednim przykładzie CPU jest bezczynne w okresie 0-1, mimo gotowości zadania J_1.

Notes

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.

Notes

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.

Notes

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

Notes

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;
- system zezwala na czasy bezczynności przy gotowym zadaniu;
- jednoprocesorowy
- zadanie: znajdź plan minimalizujący maksymalne opóźnienie.

Notes

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;
- system zezwala na czasy bezczynności przy gotowym zadaniu;
- jednoprocesorowy
- zadanie: znajdź plan minimalizujący maksymalne opóźnienie.

W ogólności, problem jest **NP-trudny**.

Notes

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;
- 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*.

Notes

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:**

Notes

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;

Notes

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).

Notes

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 → bieżąca lista realizowalnym rozwiązaniem;
  2. jeśli istnieje zadanie, którego *deadline* minął, a nie zostało jeszcze zaplanowane → bieżąca ścieżka to nierealizowalny plan (cofnij do ostatniego dokonanego wyboru).

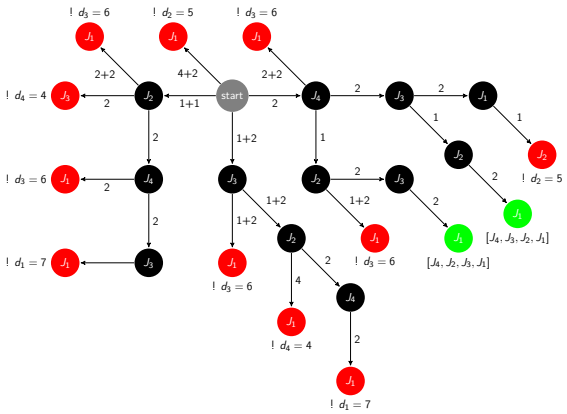
Notes

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 → bieżąca lista realizowalnym rozwiązaniem;
  2. jeśli istnieje zadanie, którego *deadline* minął, a nie zostało jeszcze zaplanowane → 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

Notes



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

Notes

Algorytm Bratley'a - własności

- Złożoność wykładnicza – tylko jako algorytm offline;

Notes

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.

Notes

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 :-)

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
- Np. dla przykładu gdzie wszystkie zadania są dostępne od razu.

Notes

Planowanie zadań z wymaganą kolejnością wykonania

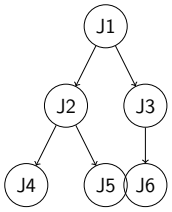
- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
- Np. dla przykładu gdzie wszystkie zadania są dostępne od razu.

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
- Np. dla przykładu gdzie wszystkie zadania są dostępne od razu.

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6



Notes