# Algorytm Horna i Sidneya

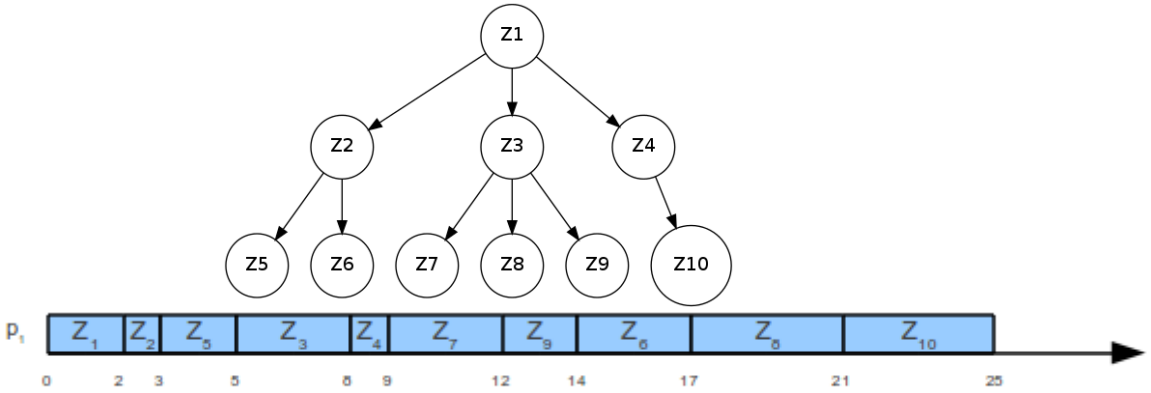
## minimalizacja średniego czasu przepływu
## jeden procesor

### zadania niepodzielne, tworzące drzewo

Jest to algorytm wielomianowy , optymalny, jego kroki przedstawiają się następująco:

1. Dla wszystkich zadań końcowych w grafie oblicz $p_j = \frac{\tau_j}{w_j}$
2. Dla zadania $Z_k$ , dla którego nie określono jeszcze współczynnika $p_k$ , a którego wszystkie następniki mają już określone wartości tego współczynnika podstaw $p_k := \frac{\tau_k}{w_k}$
3. Wyznacz zadanie, powiedzmy $Z_i$ , o minimalnej wartości współczynnika $p_k$, będace bezpośrednim nastepnikiem zadania $Z_k$. Jeśli $p_k > p_i$ *, podstaw $\tau_k := \tau_k + \tau_i'$ , $w_k := w_k + w_i', p_k := \frac{\tau_k}{w_k}$ , gdzie $\tau_i'$ jest suma czasów wykonywania, a $w_i$ jest sumą priorytetów zadań branych pod uwagę przy obliczaniu wskaźnika pi dla zadania $Z_i$. Skreśl zadanie $Z_i$ z listy bezpośrednich następników zadania $Z_k$ i powtórz krok3. W przeciwnym razie * przejdź do kroku 4.
4. jeśli nie wszystkim zadaniom przydzielono współczynniki $p_j$, to przejdź do kroku 2. W przeciwnym razie skonstruuj uszeregowanie optymalne dla wyjściowego grafu, wykonując zadania o kolejności nie malejących wartości współczynnika $p_j$, przestrzegając przy tym ograniczeń kolejnościowych.

Przykład szeregowania zbioru zadań według algorytmu Horna i Sidneya
$n$=10, $\tau$=[2,1,3,1,2,3,3,4,2,4], $w$=[3,4,5,1,4,1,2,1,1,1]




Jak łatwo zauważyć, złożoność obliczeniowa powyższego algorytmu jest $O$(n*log(n)). Wynika to z faktu, że w algorytmie tym musimy zadania uprządkować w kolejności nie malejących wartości współczynnika $p_j$.

### zadania niepodzielne, tworzące antydrzewo

1. Zamień następniki z poprzednikami we wszystkich ograniczeniach kolejnościowych, tworząc graf typu drzewa.
2. Zastosuj algorytm Horna i Sidneya do maksymalizacji $\overline{F_w}$
3. Odwróć otrzymany ciąg zadań