- 作业讲解
  - JH第4章练习4.2.1.4、4.2.1.5、4.2.3.3、4.2.3.4、4.2.3.5

# JH第4章练习4.2.1.4

- R.L. Graham.
  Bounds for Certain Multiprocessing Anomalies.
  Bell System Technical Journal, 45(9):1563-1581, 1966.

- 另一种简单证明方法：

$$cost(\mathrm{GMS}(I)) - Opt_{\mathrm{MS}}(I) \ \le\ p_k \le Opt_{\mathrm{MS}}(I)$$

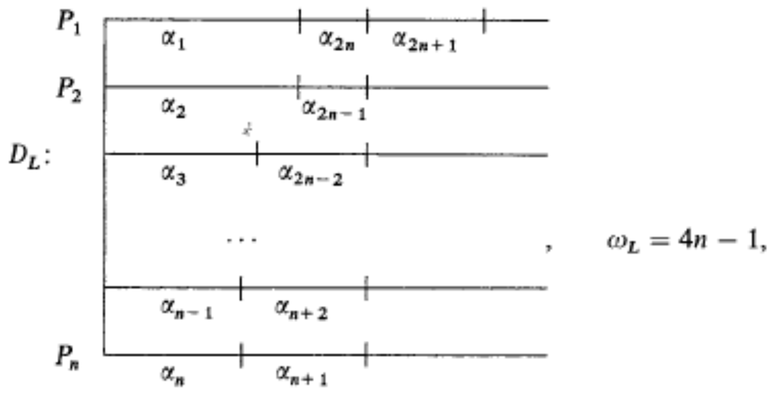
$$\frac{cost(\mathrm{GMS}(I)) - Opt_{\mathrm{MS}}(I)}{Opt_{\mathrm{MS}}(I)} \le 1$$

# JH第4章练习4.2.1.5

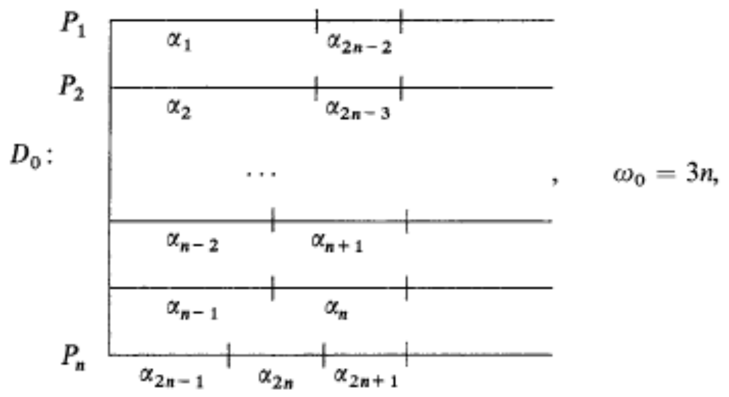
- GMS的近似比：4/3
  - R.L. Graham.
    Bounds on Multiprocessing Timing Anomalies.
    SIAM Journal on Applied Mathematics, 17(2):416-429, 1969.

To show that this bound is best possible, we consider the following set of task lengths:

$$(\alpha_1, \alpha_2, \cdots, \alpha_r) = (2n-1, 2n-1, 2n-2, 2n-2, \cdots, n+1, n+1, n, n, n),$$ 共2n+1个task



GMS的解



最优解

- 教材讨论
  - JH第4章第3节第4小节

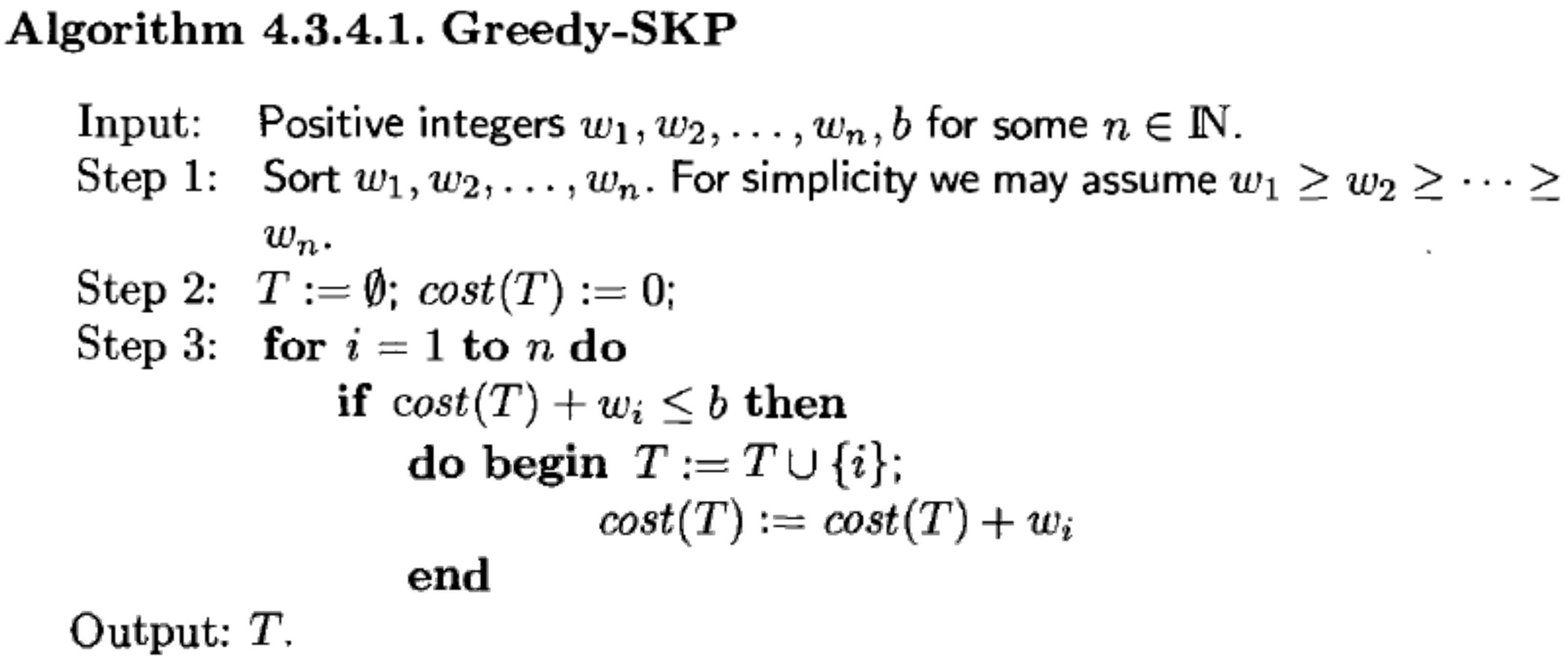
# 讨论安排

- 讨论内容
  - 4.3.4.1、4.3.4.2(用于SKP)、4.3.4.2(用于KP)、4.3.4.7、4.3.4.11
  - 算法的流程及相关证明
- 讨论分组：A、B、C、D
- 讨论流程（90分钟）
  - 15分钟：A和B组讨论4.3.4.1，C和D组讨论4.3.4.2(用于SKP)
  - 10分钟：A组上台讲解4.3.4.1，B组提问
  - 10分钟：C组上台讲解4.3.4.2(用于SKP)，D组提问

  - 15分钟：A和B组讨论4.3.4.2(用于KP)，C和D组讨论4.3.4.7
  - 10分钟：B组上台讲解4.3.4.2(用于KP)，A组提问
  - 10分钟：D组上台讲解4.3.4.7，C组提问

  - 10分钟：所有组讨论4.3.4.11
  - 10分钟：？组上台讲解4.3.4.11

# 算法4.3.4.1



**Algorithm 4.3.4.1. Greedy-SKP**

```
Input:   Positive integers $w_1, w_2, \ldots, w_n, b$ for some $n \in \mathbb{N}$.
Step 1:  Sort $w_1, w_2, \ldots, w_n$. For simplicity we may assume $w_1 \geq w_2 \geq \cdots \geq$
         $w_n$.
Step 2:  $T := \emptyset$; $cost(T) := 0$;
Step 3:  for $i = 1$ to $n$ do
             if $cost(T) + w_i \leq b$ then
                 do begin $T := T \cup \{i\}$;
                          $cost(T) := cost(T) + w_i$
                 end
Output: $T$.
```

To see that Algorithm 4.3.4.1 is a 2-approximation algorithm for SKP it is sufficient to show that $cost(T) \geq b/2$ or $T$ is an optimal solution. Without loss of generality one can assume $b \geq w_1 \geq w_2 \geq \cdots \geq w_n$. Let $j+1$ be the smallest integer not in $T$ (i.e., $\{1, 2, \ldots, j\} \subseteq T\}$). Note, that $T \neq \emptyset$ (i.e., $j \geq 1$) because $w_1 \leq b$. If $j = 1$, then $w_1 + w_2 > b$. Since $w_1 \geq w_2$ it is obvious that $cost(T) = w_1 > \frac{b}{2}$. Thus, we may assume $j \geq 2$. In general, we have

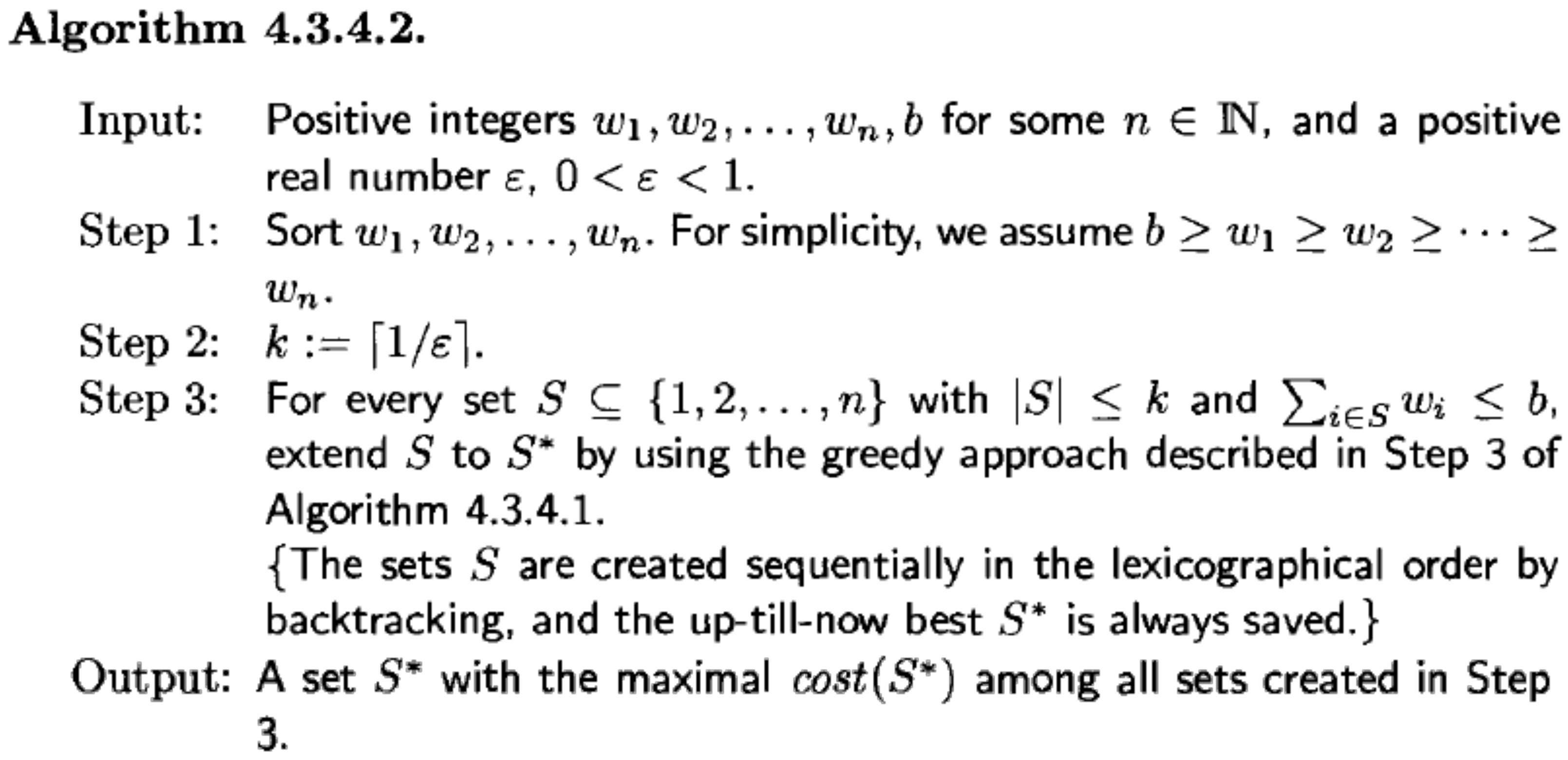
$$cost(T) + w_{j+1} > b \geq Opt_{\text{SKP}}(w_1, \ldots, w_n, b).$$

Since $w_1 \geq w_2 \geq \cdots \geq w_n$,

$$w_{j+1} \leq w_j \leq \frac{w_1 + w_2 + \cdots + w_j}{j} \leq \frac{b}{j}. \tag{4.19}$$

Thus, $cost(T) > b - w_{j+1} \geq b - \frac{b}{j} \geq b/2$ for every integer $j \geq 2$.

## 算法4.3.4.2(用于SKP)



**Algorithm 4.3.4.2.**

Input: Positive integers $w_1, w_2, \ldots, w_n, b$ for some $n \in \mathbb{N}$, and a positive real number $\varepsilon$, $0 < \varepsilon < 1$.

Step 1: Sort $w_1, w_2, \ldots, w_n$. For simplicity, we assume $b \geq w_1 \geq w_2 \geq \cdots \geq w_n$.

Step 2: $k := \lceil 1/\varepsilon \rceil$.

Step 3: For every set $S \subseteq \{1, 2, \ldots, n\}$ with $|S| \leq k$ and $\sum_{i \in S} w_i \leq b$, extend $S$ to $S^*$ by using the greedy approach described in Step 3 of Algorithm 4.3.4.1.
{The sets $S$ are created sequentially in the lexicographical order by backtracking, and the up-till-now best $S^*$ is always saved.}

Output: A set $S^*$ with the maximal $cost(S^*)$ among all sets created in Step 3.

Now, we show that the approximation ratio

$$R_{\text{Algorithm 4.3.4.2}}(I, \varepsilon) \leq 1 + \frac{1}{k} \leq 1 + \varepsilon$$

for every input $I = w_1, w_2, \ldots, w_n, b$ of SKP. Let $M = \{i_1, i_2, \ldots, i_p\}$, $i_1 < i_2 < \cdots < i_p$ be an optimal solution for $I$, i.e., $cost(M) = Opt_{\text{SKP}}(I)$. We distinguish two possibilities according to the relation between $p$ and $k = \lceil 1/\varepsilon \rceil$.

If $p \leq k$, then Algorithm 4.3.4.2 considers the set $M$ in Step 3 and so $S^*$ is an optimal solution (i.e., the relative error is 0).

Let $p > k$. Algorithm 4.3.4.2 extends the set $P = \{i_1, i_2, \ldots, i_k\}$ containing the indices of the $k$ greatest weights $w_1, w_2, \ldots, w_{i_k}$ involved in $cost(M) = w_{i_1} + \cdots + w_{i_p}$. If $P^* = M$, then we are ready. If $P^* \neq M$, then there exists $i_q \in M - P^*$ such that $i_q > i_k \geq k$ and

$$cost(P^*) + w_{i_q} > b \geq cost(M). \tag{4.20}$$

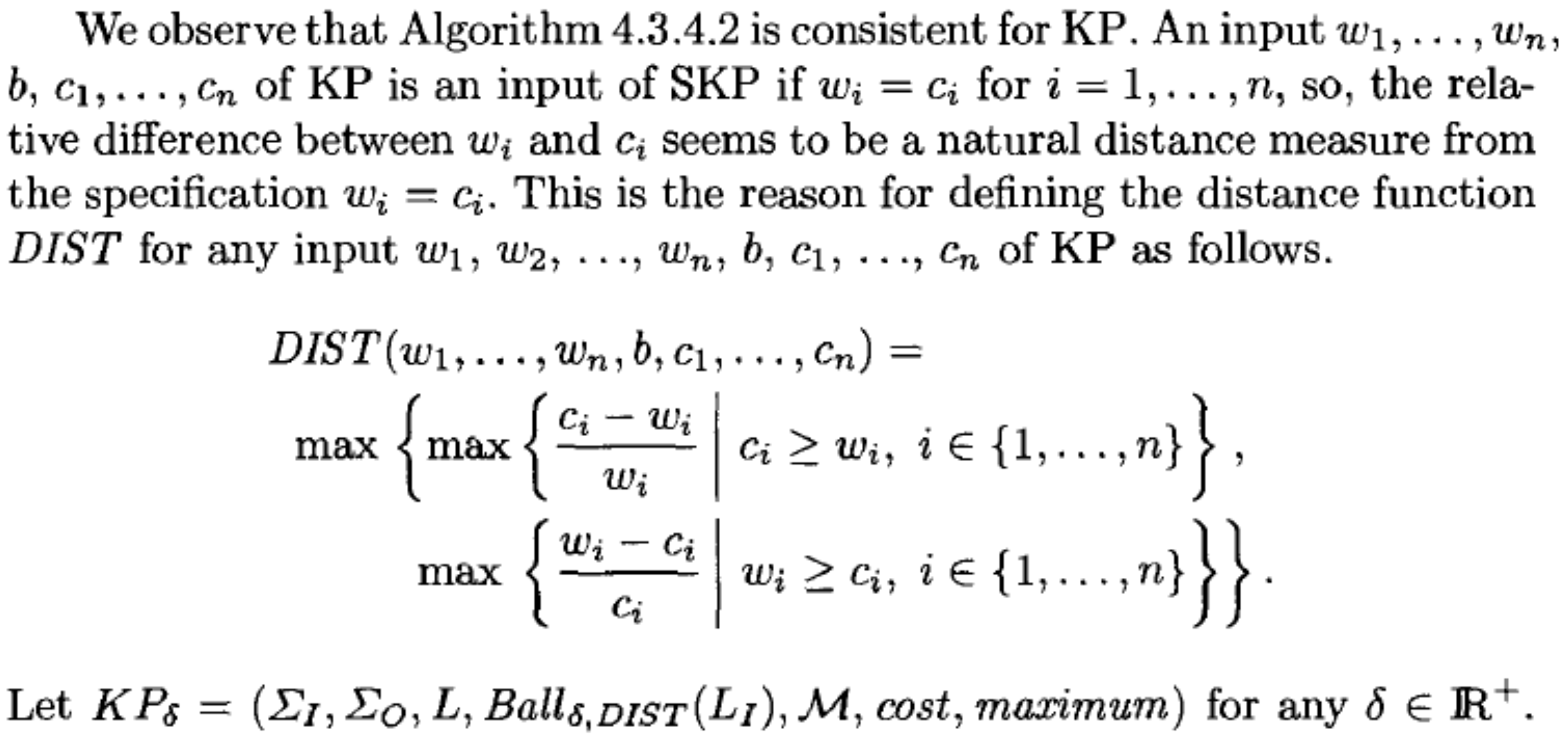
Since

$$w_{i_q} \leq \frac{w_{i_1} + w_{i_2} + \cdots + w_{i_k} + w_{i_q}}{k+1} \leq \frac{cost(M)}{k+1} \tag{4.21}$$

we obtain

$$\begin{aligned} R(I, \varepsilon) &= \frac{cost(M)}{cost(S^*)} \leq \frac{cost(M)}{cost(P^*)} \underset{(4.20)}{\leq} \frac{cost(M)}{cost(M) - w_{i_q}} \\ &\underset{(4.21)}{\leq} \frac{cost(M)}{cost(M) - \frac{cost(M)}{k+1}} = \frac{1}{1 - \frac{1}{k+1}} = \frac{k+1}{k} \\ &= 1 + \frac{1}{k} \leq 1 + \varepsilon. \end{aligned}$$

# 算法4.3.4.2(用于KP)



We observe that Algorithm 4.3.4.2 is consistent for KP. An input $w_1, \ldots, w_n$, $b$, $c_1, \ldots, c_n$ of KP is an input of SKP if $w_i = c_i$ for $i = 1, \ldots, n$, so, the relative difference between $w_i$ and $c_i$ seems to be a natural distance measure from the specification $w_i = c_i$. This is the reason for defining the distance function $DIST$ for any input $w_1$, $w_2$, …, $w_n$, $b$, $c_1$, …, $c_n$ of KP as follows.

$$\begin{aligned} & DIST(w_1, \ldots, w_n, b, c_1, \ldots, c_n) = \\ & \max \left\{ \max \left\{ \frac{c_i - w_i}{w_i} \,\middle|\, c_i \geq w_i,\ i \in \{1, \ldots, n\} \right\}, \right. \\ & \qquad \left. \max \left\{ \frac{w_i - c_i}{c_i} \,\middle|\, w_i \geq c_i,\ i \in \{1, \ldots, n\} \right\} \right\}. \end{aligned}$$

Let $KP_\delta = (\Sigma_I, \Sigma_O, L, Ball_{\delta, DIST}(L_I), \mathcal{M}, cost, maximum)$ for any $\delta \in \mathbb{R}^+$.

Let us consider $\{\text{ASKP}_\varepsilon\}_{\varepsilon>0}$ as the collection of $(1+\varepsilon)$-approximation algorithms determined by Algorithm 4.3.4.2.

**Lemma 4.3.4.5.** *For every $\varepsilon > 0$ and every $\delta > 0$, the algorithm* $\text{ASKP}_\varepsilon$ *is a $(1+\delta^2+\varepsilon+\varepsilon\cdot\delta^2)$-approximation algorithm for* $\text{KP}_\delta$.

*Proof.* Let $I = (w_1, w_2, \ldots, w_n, b, c_1, c_2, \ldots, c_n)$ be an input instance of $\text{KP}_\delta$, i.e.,

$$(1+\delta)^{-1} \le \frac{w_i}{c_i} \le 1+\delta \tag{4.24}$$

for all $i = 1, 2, \ldots, n$.

Let $U$ be an optimal solution of $I$ and let $T^*$ be the output of $\text{ASKP}_\varepsilon$ for $I$. Since $\text{ASKP}_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm for $I' = (w_1, w_2, \ldots, w_n, b)$ of $\text{SKP}_\varepsilon$, we have

$$\frac{\sum_{i\in U} w_i}{\sum_{j\in T^*} w_j} \le 1+\varepsilon. \tag{4.25}$$

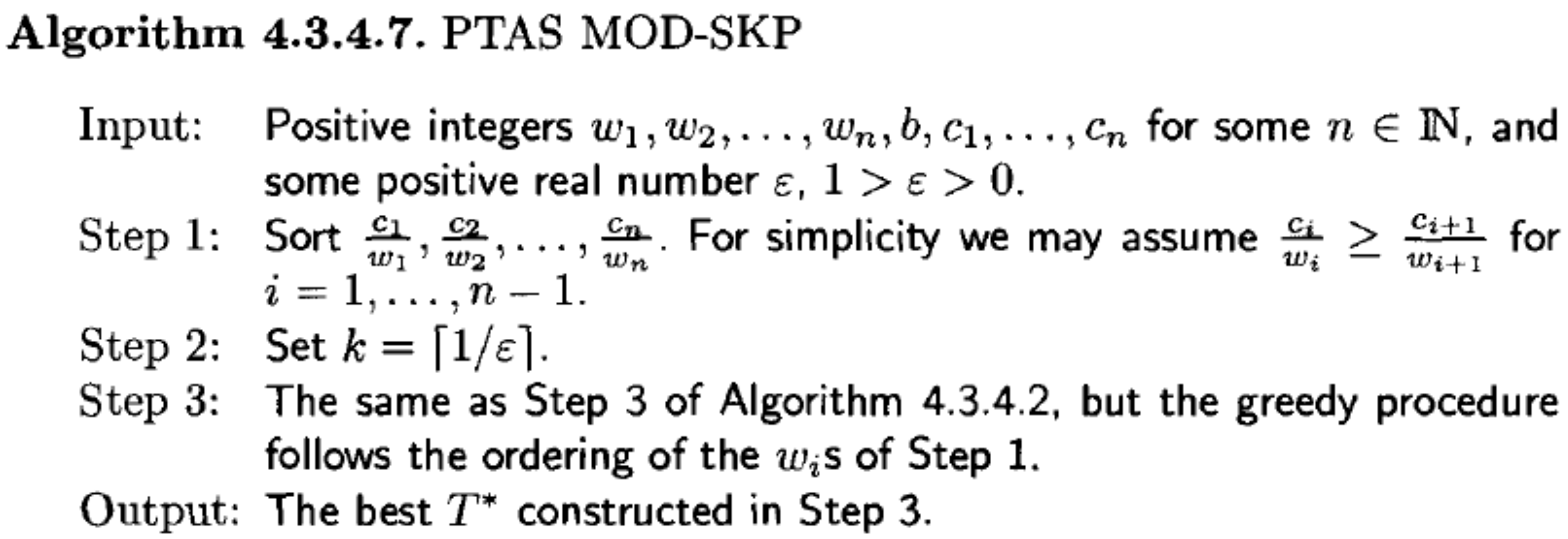
Now, we are ready to estimate the approximation ratio of $\text{ASKP}_\varepsilon$ for the input instance $I$.

$$\begin{aligned} R(I,\varepsilon) = \frac{cost(U)}{cost(T^*)} &\underset{(4.24)}{\le} \frac{\sum_{i\in U} w_i \cdot (1+\delta)}{\sum_{j\in T^*} w_j \cdot (1+\delta)^{-1}} \\ &= (1+\delta)^2 \cdot \frac{\sum_{i\in U} w_i}{\sum_{j\in T^*} w_j} \underset{(4.25)}{\le} (1+\delta)^2\cdot(1+\varepsilon) \\ &= 1+\delta^2+\varepsilon+\varepsilon\cdot\delta^2. \end{aligned}$$

为什么不是superstable？

为什么不是$\text{KP}_\delta$的PTAS？

# 算法4.3.4.7



**Algorithm 4.3.4.7.** PTAS MOD-SKP

Input: Positive integers $w_1, w_2, \ldots, w_n, b, c_1, \ldots, c_n$ for some $n \in \mathbb{N}$, and some positive real number $\varepsilon$, $1 > \varepsilon > 0$.

Step 1: Sort $\frac{c_1}{w_1}, \frac{c_2}{w_2}, \ldots, \frac{c_n}{w_n}$. For simplicity we may assume $\frac{c_i}{w_i} \geq \frac{c_{i+1}}{w_{i+1}}$ for $i = 1, \ldots, n-1$.

Step 2: Set $k = \lceil 1/\varepsilon \rceil$.

Step 3: The same as Step 3 of Algorithm 4.3.4.2, but the greedy procedure follows the ordering of the $w_i$s of Step 1.

Output: The best $T^*$ constructed in Step 3.

**Lemma 4.3.4.8.** *For every $\varepsilon$, $1 > \varepsilon > 0$ and every $\delta \geq 0$,* MOD-SKP$_\varepsilon$ *is a* $(1 + \varepsilon \cdot (1 + \delta) \cdot (1 + \varepsilon))$*-approximation algorithm for* KP$_\delta$.

*Proof.* Let $U = \{i_1, i_2, \ldots, i_l\} \subseteq \{1, 2, \ldots, n\}$, where $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_l}$[13] be an optimal solution for the input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$.

If $l \leq k$, then MOD-SKP$_\varepsilon$ provides an optimal solution.

If $l > k$, then we consider a $T^* = \{i_1, i_2, \ldots, i_k, j_{k+1}, \ldots, j_{k+r}\}$ as a greedy extension of $T = \{i_1, i_2, \ldots, i_k\}$. Again, we distinguish two possibilities according to the sizes of $\sum_{i \in U} w_i$ and $\sum_{j \in T^*} w_j$.

(i) Let $\sum_{i \in U} w_i - \sum_{j \in T^*} w_j < 0$.
Now, we show that this is impossible because it contradicts the optimality of $U$. Both $cost(U)$ and $cost(T^*)$ contain $\sum_{s=1}^{k} c_{i_s}$. For the rest $T^*$ contains the best choice of $w_i$s according to the cost of one weight unit. The choice of $U$ per one weight unit cannot be better. Thus, $cost(U) < cost(T^*)$.

(ii) Let $d = \sum_{i \in U} w_i - \sum_{j \in T^*} w_j \geq 0$.
Because of the optimal choice of $T^*$ according to the cost per one weight unit, the cost $c$ of the first part of $U$ with the weight $\sum_{j \in T^*} w_j$ is at most $cost(T^*)$, i.e.,

$$c - cost(T^*) \leq 0 \tag{4.26}$$

Since $U$ and $T^*$ contain the same $k$ indices $i_1, i_2, \ldots, i_k$, and $w_{i_1}, \ldots, w_{i_k}$ are the largest weights in both $U$ and $T^*$, the same consideration as in the proof of Lemma 4.3.4.4 yields (see (4.23))

$$d \leq \varepsilon \cdot \sum_{i \in U} w_i, \text{ and } cost(U) \leq c + d \cdot (1 + \delta). \tag{4.27}$$

Considering $U$ different from $T^*$, there exists an $m \in \{k+1, \ldots, l\}$ such that

$$i_m \in U - T^* \text{ and } \sum_{j \in T^*} w_j + w_{i_m} > b \geq \sum_{i \in U} w_i.$$

Therefore, MOD-SKP$_\varepsilon$ is also an $\varepsilon$-approximation algorithm[14] for SKP and so

$$\frac{\sum_{i \in U} w_i}{\sum_{j \in T^*} w_j} \leq 1 + \varepsilon. \tag{4.28}$$

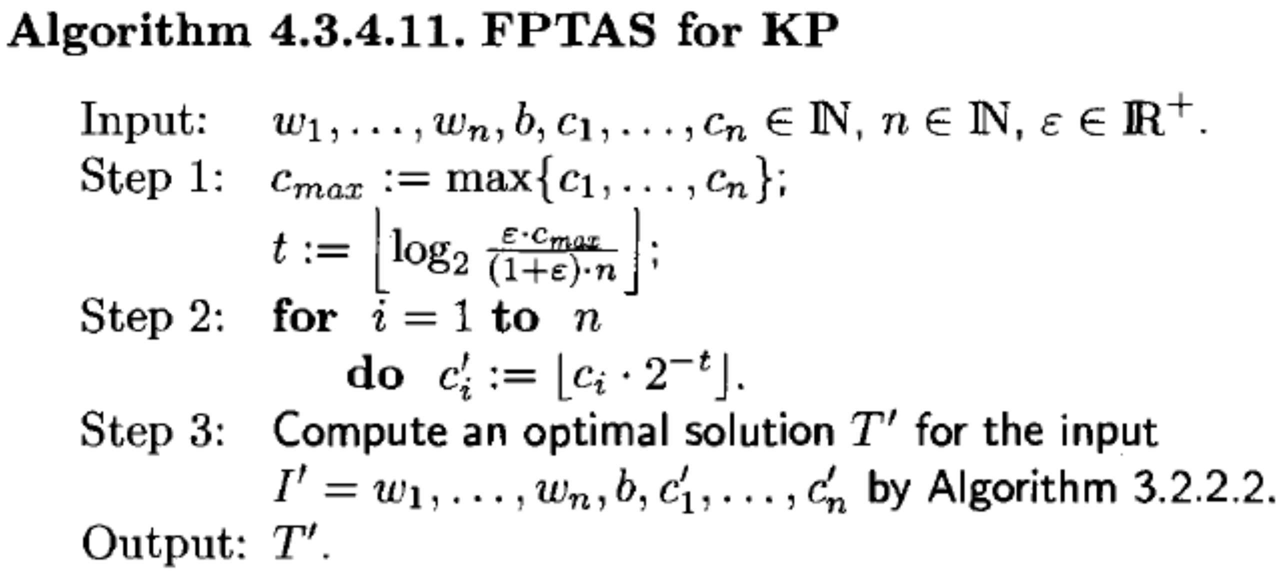
Hence,

$$\begin{aligned}
R(I, \varepsilon) &= \frac{cost(U)}{cost(T^*)} \\
&\underset{(4.27)}{\leq} \frac{c + d \cdot (1 + \delta)}{cost(T^*)} \\
&= \frac{cost(T^*) + c - cost(T^*) + d \cdot (1 + \delta)}{cost(T^*)} \\
&= 1 + \frac{c - cost(T^*) + d \cdot (1 + \delta)}{cost(T^*)} \\
&\underset{(4.26)}{\leq} 1 + \frac{d \cdot (1 + \delta)}{cost(T^*)} \\
&\underset{(4.27)}{\leq} 1 + \varepsilon \cdot (1 + \delta) \cdot \frac{\sum_{i \in U} w_i}{\sum_{j \in T^*} w_j} \\
&\underset{(4.28)}{\leq} 1 + \varepsilon \cdot (1 + \delta) \cdot (1 + \varepsilon).
\end{aligned}$$

为什么是superstable？

为什么是KP的PTAS？

# 算法4.3.4.11



**Algorithm 4.3.4.11. FPTAS for KP**

```
Input:   $w_1, \ldots, w_n, b, c_1, \ldots, c_n \in \mathbb{N}$, $n \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$.
Step 1:  $c_{max} := \max\{c_1, \ldots, c_n\}$;
         $t := \left\lfloor \log_2 \frac{\varepsilon \cdot c_{max}}{(1+\varepsilon) \cdot n} \right\rfloor$;
Step 2:  for $i = 1$ to $n$
             do $c_i' := \lfloor c_i \cdot 2^{-t} \rfloor$.
Step 3:  Compute an optimal solution $T'$ for the input
         $I' = w_1, \ldots, w_n, b, c_1', \ldots, c_n'$ by Algorithm 3.2.2.2.
Output:  $T'$.
```

To obtain an FPTAS for KP we consider a completely new approach. In Section 3.2 we presented Algorithm 3.2.2.2 (based on dynamic programming) for KP working in time $O(n \cdot Opt_{\mathrm{KP}}(I))$ for any input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$. The trouble is that $Opt_{\mathrm{KP}}(I)$ can be exponential in the input length and so the time complexity of Algorithm 3.2.2.2 has to be considered to be exponential. The idea of our FPTAS for KP is to "approximate" every input $I = w_1, \ldots, w_n, b, c_1, \ldots, c_n$ by another input with $I' = w_1, w_2, \ldots, w_n, b, c'_1, \ldots, c'_n$ with $\sum_{i=1}^{n} c'_i$ polynomial in $n$, and then to apply Algorithm 3.2.2.2 to the input $I'$ in order to get a feasible solution for $I$. If one wants to obtain $Opt_{\mathrm{KP}}(I')$ to be small in $n$, then one has to divide the input values $c_1, \ldots, c_n$ by a same large number $d$. Obviously, the efficiency increases with growing $d$, but the approximation ratio increases with growing $d$, too. So, $d$ may be chosen dependent on the user preference.

**Theorem 4.3.4.12.** *Algorithm 4.3.4.11 is an* FPTAS *for* KP.

*Proof.* First, we show that Algorithm 4.3.4.11 is an approximation scheme. Since the output $T'$ (an optimal solution for $I'$) is a feasible solution for $I'$, and $I$ and $I'$ do not differ in the weights $w_1, \ldots, w_n, b$, $T'$ is a feasible solution for $I$, too. Let $T$ be an optimal solution for the original input $I$. Our goal is to show that

$$R(I) = \frac{cost(T,I)}{cost(T',I)} \leq 1 + \varepsilon.$$

We have:

$$\begin{aligned}
cost(T,I) &= \sum_{j \in T} c_j \\
&\geq \sum_{j \in T'} c_j = cost(T',I) \quad \{\text{because } T \text{ is optimal for } I \\
&\qquad\qquad\qquad\qquad\qquad\quad \text{and } T' \text{ is feasible for } I\} \\
&\geq 2^t \cdot \sum_{j \in T'} c'_j \quad \{\text{follows from } c'_j = \lfloor c_j \cdot 2^{-t} \rfloor\} \\
&\geq 2^t \sum_{j \in T} c'_j \quad \{\text{because } T' \text{ is optimal for } I'\} \\
&= \sum_{j \in T} 2^t \cdot \lfloor c_j \cdot 2^{-t} \rfloor \quad \{\text{because } c'_j = \lfloor c_j \cdot 2^{-t} \rfloor\} \\
&\geq \sum_{j \in T} 2^t \left(c_j \cdot 2^{-t} - 1\right) \geq \left(\sum_{j \in T} c_j\right) - n \cdot 2^t = cost(T,I) - n \cdot 2^t.
\end{aligned}$$

We have, thus, proved

$$cost(T,I) \geq cost(T',I) \geq cost(T,I) - n \cdot 2^t, \text{ i.e.,} \tag{4.29}$$

$$\begin{aligned}
0 &\leq cost(T,I) - cost(T',I) \leq n \cdot 2^t \\
&\leq n \cdot \frac{\varepsilon \cdot c_{max}}{(1+\varepsilon) \cdot n} = \varepsilon \cdot \frac{c_{max}}{1+\varepsilon}.
\end{aligned} \tag{4.30}$$

Since we may assume $cost(T,I) \geq c_{max}$ ($w_i \leq b$ for every $i = 1, \ldots, n$), we obtain from (4.29) and (4.30)

$$cost(T', I) \geq c_{max} - \varepsilon \cdot \frac{c_{max}}{1+\varepsilon}. \tag{4.31}$$

Finally,

$$\begin{aligned} R(I) = \frac{cost(T,I)}{cost(T',I)} &= \frac{cost(T',I) + cost(T,I) - cost(T',I)}{cost(T',I)} \\ &\leq 1 + \frac{\varepsilon \cdot \frac{c_{max}}{1+\varepsilon}}{cost(T',I)} \quad \{\text{because of } (4.30)\} \\ &\leq 1 + \frac{\varepsilon \cdot \frac{c_{max}}{1+\varepsilon}}{c_{max} - \varepsilon \cdot \frac{c_{max}}{1+\varepsilon}} \quad \{\text{because of } (4.31)\} \\ &= 1 + \frac{\varepsilon}{1+\varepsilon} \cdot \frac{1}{1 - \frac{\varepsilon}{1+\varepsilon}} = 1 + \frac{\varepsilon}{1+\varepsilon} \cdot (1+\varepsilon) = 1 + \varepsilon. \end{aligned}$$

Now, we have to prove that the time complexity of Algorithm 4.3.4.11 is polynomial in $n$ and $\varepsilon^{-1}$. Step 1 and Step 2 can be executed in time $O(n)$. Step 3 is the run of Algorithm 3.2.2.2 on the input $I'$ and this can be performed in time $O(n \cdot Opt_{\text{KP}}(I'))$. We bound $Opt_{\text{KP}}(I')$ as follows:

$$\begin{aligned} Opt_{\text{KP}}(I') \leq \sum_{i=1}^{n} c'_i &= \sum_{i=1}^{n} \left\lfloor c_i \cdot 2^{-\left\lfloor \log_2 \frac{\varepsilon \cdot c_{max}}{(1+\varepsilon) \cdot n} \right\rfloor} \right\rfloor \\ &\leq \sum_{i=1}^{n} \left( c_i \cdot 2 \cdot \frac{(1+\varepsilon) \cdot n}{\varepsilon \cdot c_{max}} \right) \\ &= 2 \cdot (1+\varepsilon) \cdot \varepsilon^{-1} \cdot \frac{n}{c_{max}} \cdot \sum_{i=1}^{n} c_i \\ &\leq 2 \cdot (1+\varepsilon) \cdot \varepsilon^{-1} \cdot n^2 \in O\left(\varepsilon^{-1} \cdot n^2\right). \end{aligned}$$

Thus, Algorithm 4.3.4.11 works in time $O\left(\varepsilon^{-1} \cdot n^3\right)$. □