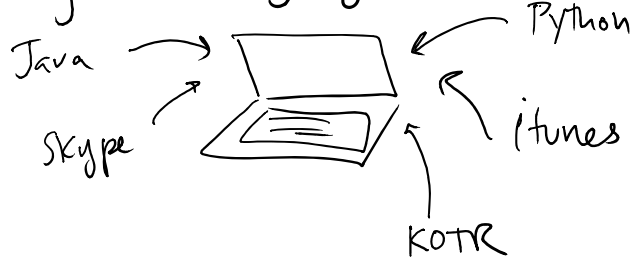
S. KIMMEL

Scheduling

Suppose you are trying to run many applications on a processor



What order is best?

Each job i ; $i \in \{1, \ldots, n\}$

- weight (importance) w_i
- time t_i to complete

def: Completion time C_j of job j is sum of times required to complete all jobs run before j, plus t_j.

Q: Suppose there are 3 jobs, with $t_1 = 1$, $t_2 = 2$, $t_3 = 3$, and that they are run in reverse order. What is C_1, C_2, C_3

A) 3, 2, 1 B) 1, 2, 3 C) 3, 6, 7

D) 3, 5, 6

S. KIMMEL

Scheduling Goal:

Minimize $\sum_{j=1}^{n} w_j C_j$ $\Leftarrow$ "Objective function" A

Q: Why is this a good goal?

A: If important jobs left until end, $w_j C_j \rightarrow$ A large

large large

ex:

job	time	weight
1	1	3
2	2	2
3	3	1

Order	A
123	
132	
213	
231	
312	
321	

S. KIMMEL

Greedy algorithm: picks best job to schedule first

$\Rightarrow$ What is best? Create function $f(w_i, t_i)$, find job with largest f-value. Put first. Repeat!

Q: Create a good f:

A: Good jobs have large weight, short time

- $f_1 = \frac{w_i}{t_i}$
- $f_2 = w_i - t_i$
- $f = \frac{w_i}{t_i} + w_i - t_i$

Which to Choose!? :(

Try to find simple examples where behave differently!

job	weight	time	f_1	f_2
1	1	3	$1/3 = 2/6$	-2
2	2	5	$2/5$	-3

If follow $f_1 \rightarrow$ order $(2, 1)$

If follow $f_2 \rightarrow$ order $(1, 2)$

order	A
1 2	$1 \cdot 3 + 2 \cdot 8 = 19$
2 1	$2 \cdot 5 + 1 \cdot 8 = 18$

f_1 is better!

In fact, f_1 is optimal!

S. KIMMEL

Thm: Greedy algorithm choosing by w_i/t_i is optimal.

Pf: EXCHANGE argument & Pf by Contradiction

Assume
- w_i/t_i are distinct $\forall\ i \in \{1, 2, \ldots, n\}$
- $w_1/t_1 > w_2/t_2 > \cdots > w_n/t_n$ (rename)

Let σ be ordering using greedy
Let σ^* be optimal ordering.
Suppose for contradiction, σ^* is better than σ

Note: $\exists\ k, j$ s.t. $w_j/t_j > w_k/t_k$, but j is immediately after k in σ^* ordering.
(Otherwise, $\sigma^* = \sigma$!)

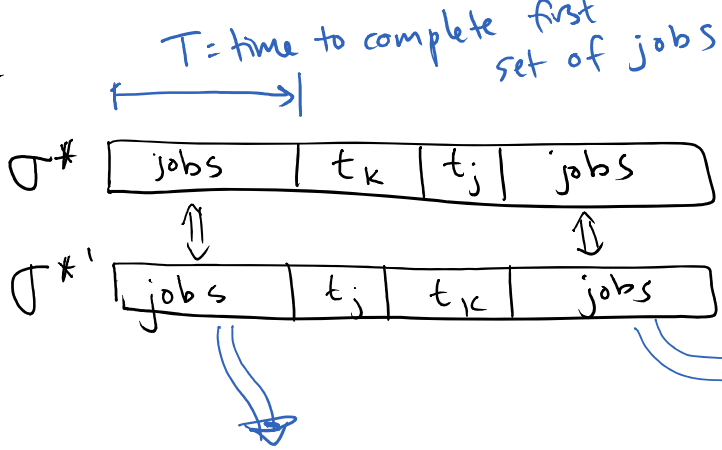
Let's create a new ordering $\sigma^{*'}$ that is same as σ^*, but with k, j positions switched

Q: If σ^* has objective value A_{σ^*}, and $\sigma^{*'}$ has objective value $A_{\sigma^{*'}}$, how are $A_{\sigma^*}, A_{\sigma^{*'}}$ related?

$$A_{\sigma^*} = A_{\sigma^{*'}} + \underline{X}$$

S. KIMMEL

A



$$A_{\sigma^*} = \sum w_i C_i + w_k(T + t_k) + w_j(T + t_k + t_j) + \sum w_r C_r$$

$$A_{\sigma^{*'}} = \sum w_i C_i + w_j(T + t_j) + w_k(T + t_j + t_k) + \sum w_r C_r$$

$$A_{\sigma^*} = B + w_j t_k \qquad \text{where } B = \sum w_i C_i + \sum w_r C_r + w_j(t_j + T) + w_k(t_k + T)$$

$$A_{\sigma^{*'}} = B + w_k t_j$$

because $t_j, t_k > 0$

$$\frac{w_j}{t_j} > \frac{w_k}{t_k} \Rightarrow w_j t_k > w_k t_j$$

$A_{\sigma^{*'}} < A_{\sigma^*} \Rightarrow$ Contradiction!

This is called an exchange argument. Show that a small alteration in a solution leads to a better solution.

What is runtime: $O(n \log n)$ (need to sort n items)

S. KIMMEL

Now let's get rid of the assumption w_i/t_i distinct $\forall\ i \in \{1, \ldots, n\}$

Q: Why does the old proof fail?

A There might not be a unique solution

B) Objective function might not decrease from σ^* to $\sigma^{*'}$

C) We can't create an ordering such that $w_1/t_1 > w_2/t_2 > \ldots > w_n/t_n$

D) All of the above

Still use EXCHANGE argument, but now need more exchanges.

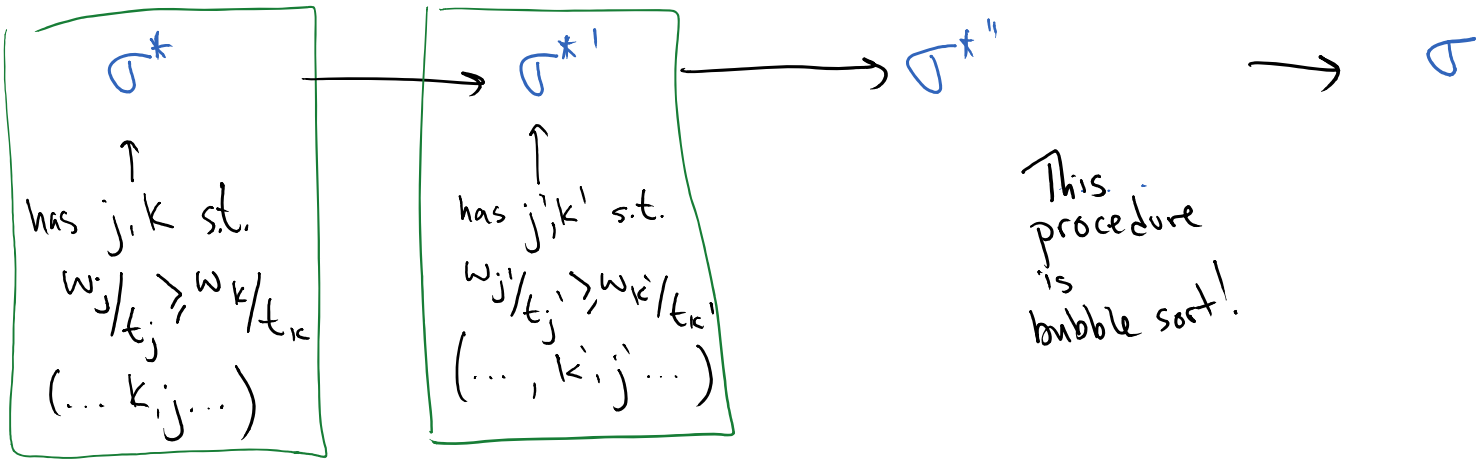
Choose some ordering s.t.

$$w_1/t_1 \geq w_2/t_2 \geq w_3/t_3 \ldots \geq w_n/t_n$$

Let σ be strategy using this ordering

Let σ^* be some other strategy

We will show $A_{\sigma^*} \geq A_\sigma$



$A_{\sigma^*} \geq A_{\sigma^{*'}} > A_{\sigma^{*''}} > A_\sigma$

S. KIMMEL

Conclusion: $A_{\sigma^*} \geq A_\sigma$, so greedy strategy is still optimal!

Q: What is run time?

A) $O(n)$ B) $O(n \log n)$ C) $O(n^2)$

D) $O(1)$

A: $O(n \log n)$. Need to sort n items, requires $O(n \log n)$ time