S. KIMMEL

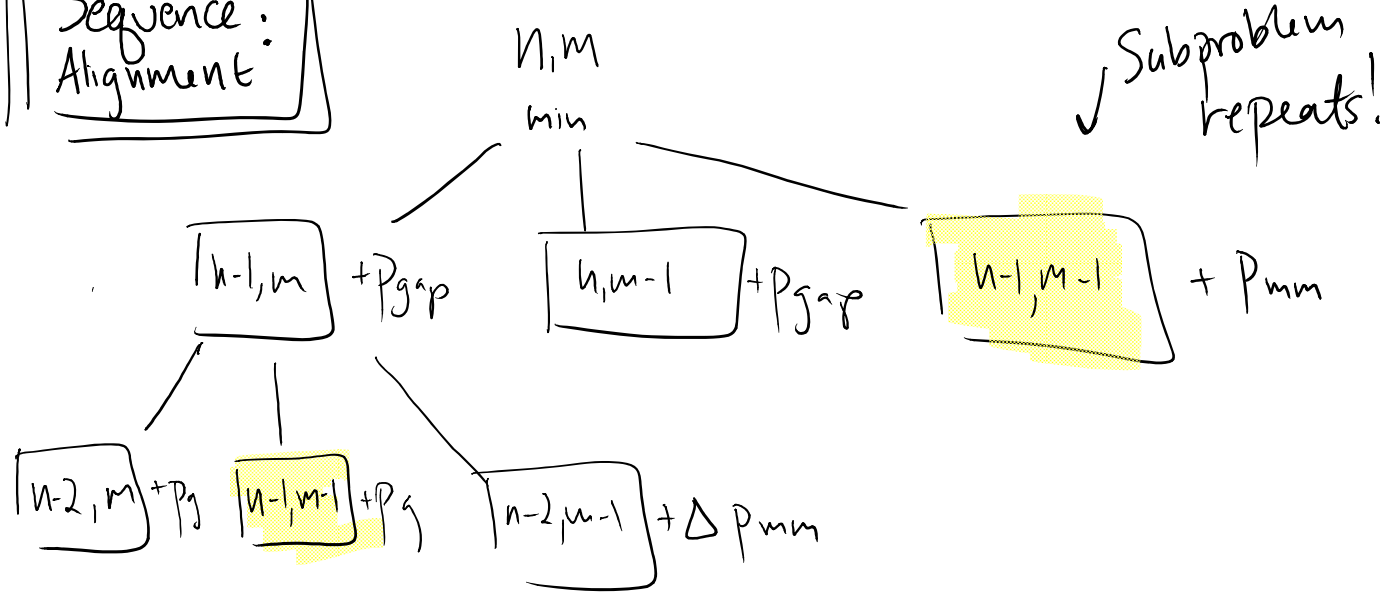
Divide & Conquer

- Solve big problem by solving subproblems
- Recursively calls same function to solve subproblems
- Often divides problem into 2 halves
- Use if subproblems don't repeat.

Dynamic

- Solve big problem by solving subproblems
- Stores subproblems in array
- Often divides into n-1, 1 (but not always)
- Use if subproblem repeats

Sequence Alignment:



S.KIMMEL

New Paradigm: Greedy Algorithm: does the thing that looks best right now

Knapsack: Capacity 10,

Items

$v_1 = 6,\ w_1 = 5$

$v_2 = 7\ w_1 = 5$

$v_3 = 10\ w_1 = 10$

Greedy algorithm would put the largest value item (#3) that fits knapsack. Not optimal!

Divide & Conquer, Dynamic Programming

- Solves subproblems
- Difficult to describe algorithm
- Relatively easy to prove correctness (induction)

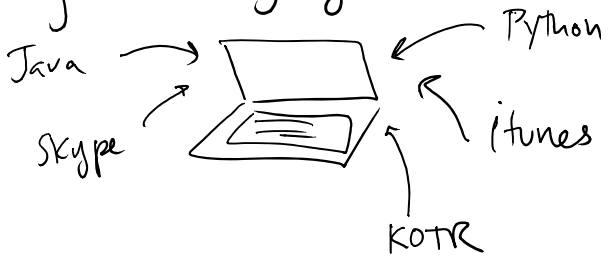
Greedy Algorithm

- Doesn't solve subproblems
- Easy to describe algorithm (usually)
- Hard to prove correctness (induction, replacement, others...)

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

Q: Why is this a good goal?

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

(w_j: large, C_j: 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

⇒ 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 → order $(2, 1)$

If follow f_2 → 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!