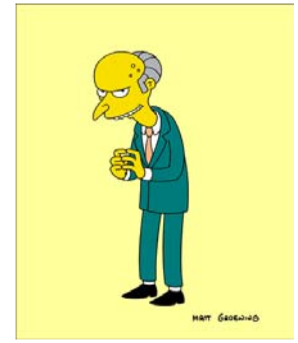
Greedy Algorithms



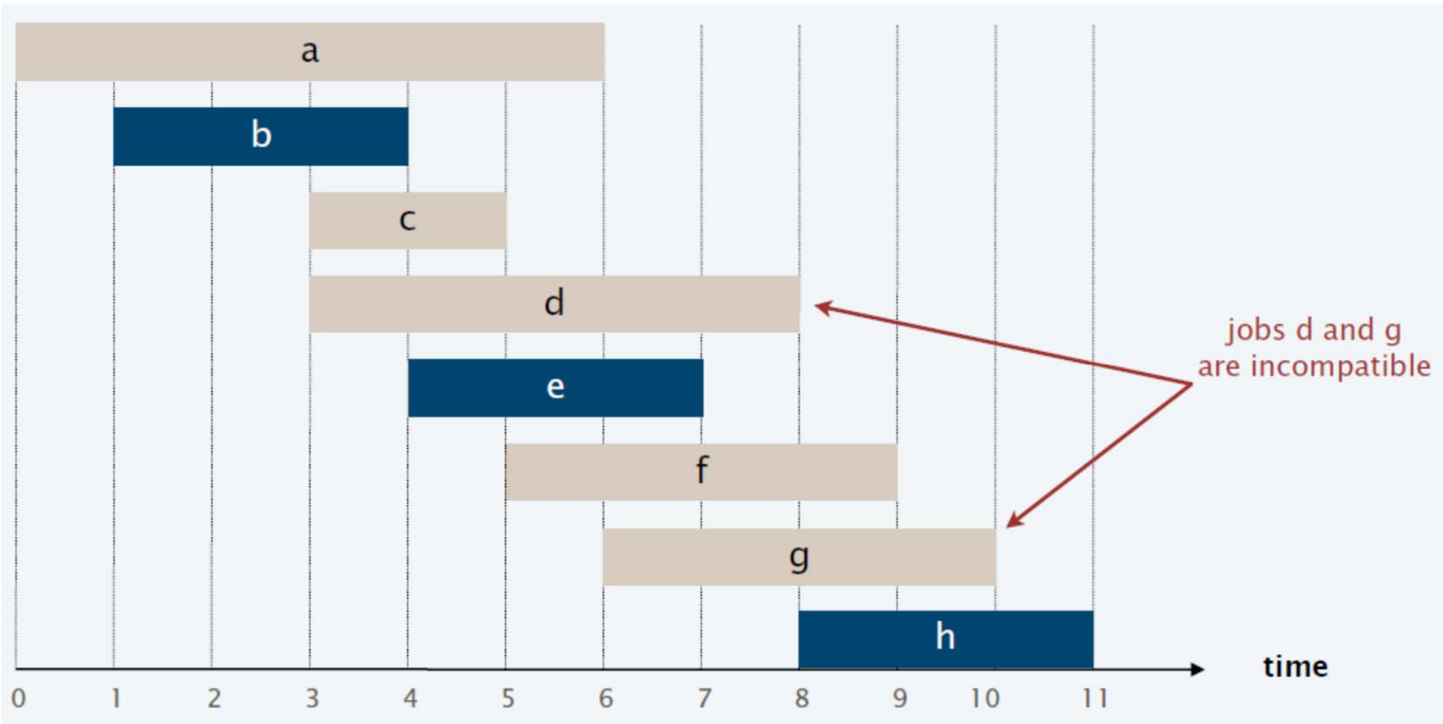
- Solve problems with the simplest possible algorithm
- The hard part: showing that something simple actually works
- Pseudo-definition
  - An algorithm is Greedy if it builds its solution by adding elements one at a time using a simple rule

Scheduling Theory

- Tasks
  - Processing requirements, release times, deadlines
- Processors
- Precedence constraints
- Objective function
  - # of jobs scheduled, delay, total execution time

Interval Scheduling

- Given a set of n tasks
- i-th task start and finish time: s(i), f(i)
- Two tasks are compatible if they don't overlap
- Goal: find the maximum number of mutually compatible tasks



Greedy template

Let T be the set of tasks, construct a set of independent tasks I

A is the rule determining the greedy algorithm

I = { }

While (T is not empty)

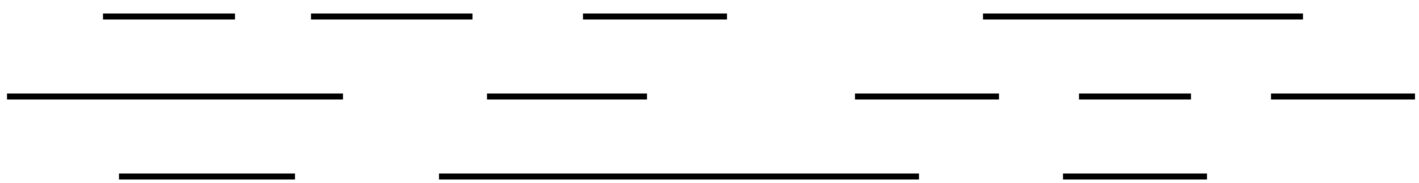
    Select a task t from T by a rule A

    Add t to I

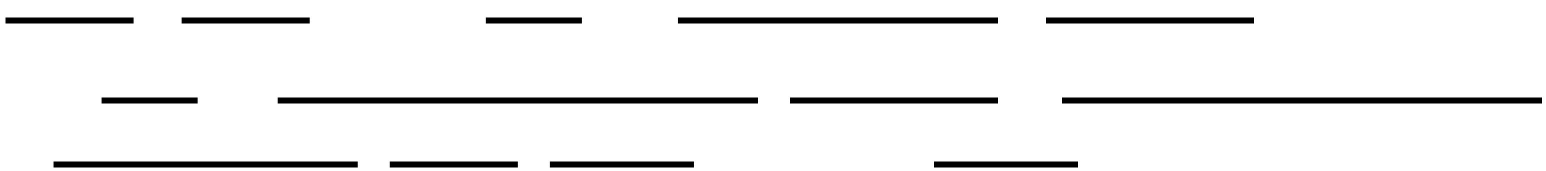
    Remove t and all tasks incompatible with t from T

Simulate the greedy algorithm for each of these heuristics

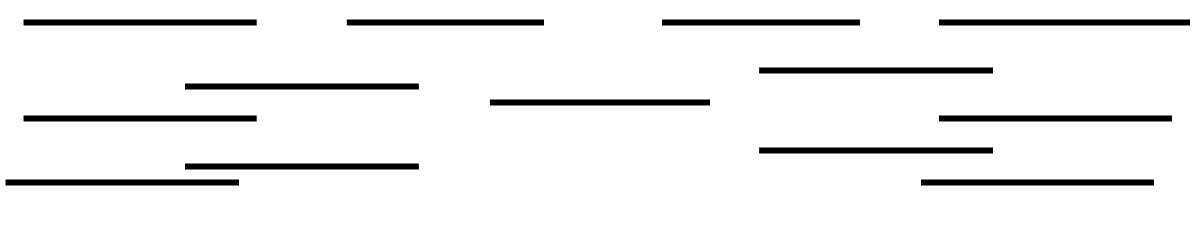
A1: Schedule earliest starting task



A2: Schedule shortest available task

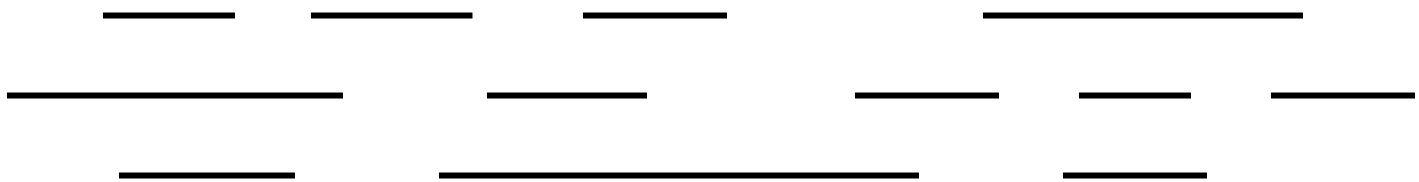


A3: Schedule task with fewest conflicting tasks

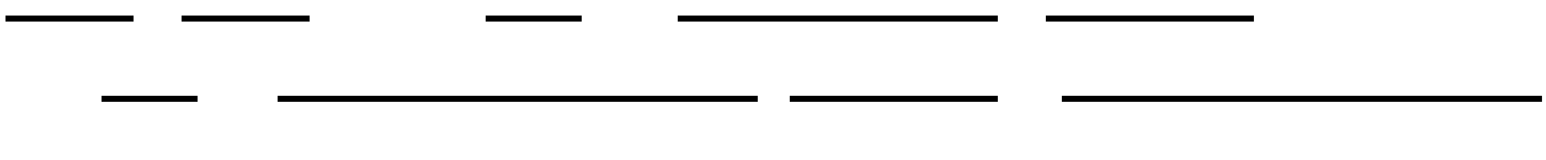


Greedy solution based on <u>earliest finishing time</u>

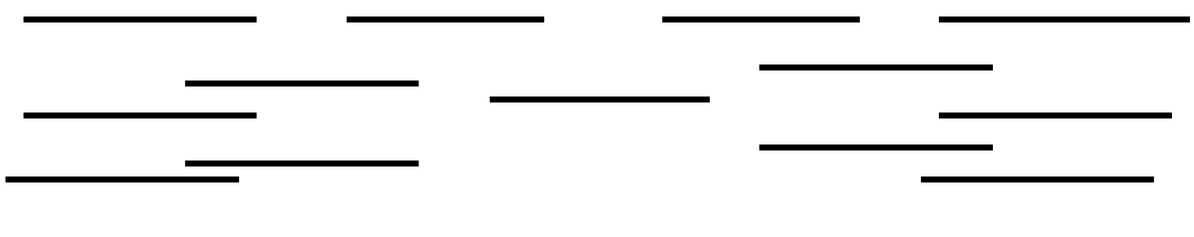
Example 1



Example 2



Example 3



```
EARLIEST-FINISH-TIME-FIRST (n,s_1,s_2,…,s_n,f_1,f_2,…,f_n)

Sort tasks by finish time so that  f_1 ≤ f_2 ≤ ⋯ ≤ f_n

A ← Φ

for i=1 to n
      if task i is compatible with A
       A ← A ∪ {i}

Return A
```

Runtime: $O(n \log n)$ – sorting

To check if task i is compatible with A, just need to keep track of the last added task j^* and compare s_i to f_{j^*}

Analysis of EARLIEST-FINISH-TIME-FIRST

Claim: the EARLIEST-FINISH-TIME-FIRST algorithm is optimal

Key idea: EARLIEST-FINISH-TIME-FIRST stays ahead

- Let $i_1\ i_2\ \ldots\ i_p$ be the set of tasks selected by greedy ordered by finish time
- Let $j_1\ j_2\ \ldots\ j_q$ be a set of tasks selected by a different algorithm ordered by finish time

We can show that for $r \le \min(p, q)$, $\boxed{f(i_r) \le f(j_r)}$

Stay ahead lemma

Proof (by induction)

<u>Base case:</u>

Greedy choose i_1 with the minimum f, so we must have $f(i_1) \leq f(j_1)$

<u>Induction assumption:</u>

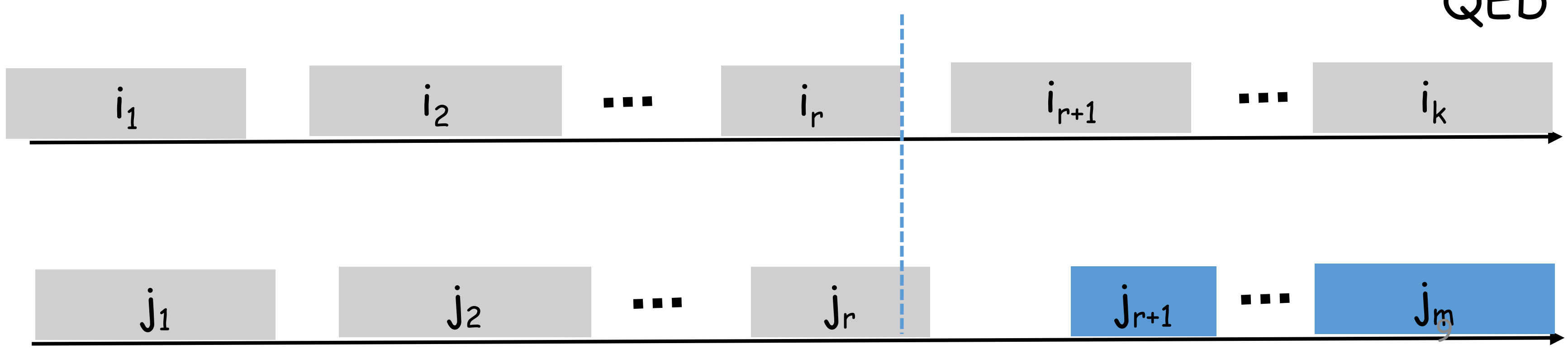
Assume this is true for $f(i_r) \leq f(j_r)$ for $r=1,\ldots,k$

<u>Inductive step:</u>

At the end of r-th iteration, task i_{r+1} and j_{r+1} are both compatible with A

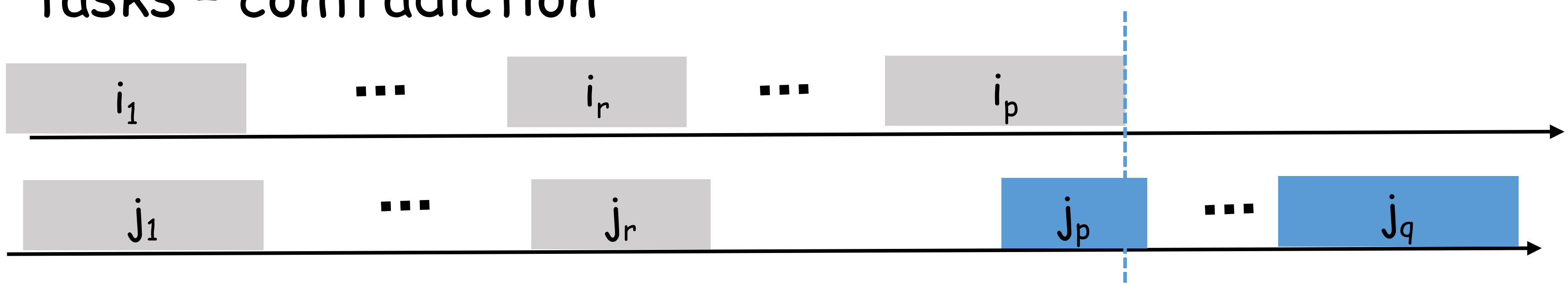
Greedy always choose the one with minimum f, thus $f(i_{r+1}) \leq f(j_{r+1})$

QED



Completing the proof

- Let $i_1, \ldots, i_p$ be the set of tasks found by EFTF in increasing order of finish times
- Let $j_1, \ldots, j_q$ be the set of tasks found by an optimal algorithm in increasing order of finish times
- If $p < q$, then the EFTF stopped before it ran out of tasks – contradiction



- Thus we must have $p=q$