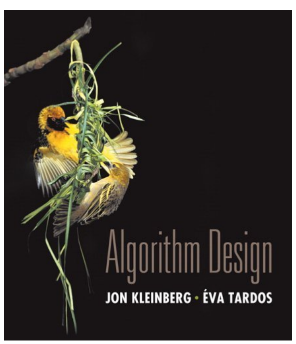
CSE 417

Chapter 4: Greedy Algorithms

START FRI JAN 18 (begins halfway thru lecture)



Many Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Greed is good. Greed is right. Greed works. Greed clarifies, cuts through, and captures the essence of the evolutionary spirit.

- Gordon Gecko (Michael Douglas)

Intro: Coin Changing

Coin Changing

Goal. Given currency denominations: 1, 5, 10, 25, 100, give change to customer using *fewest* number of coins.

Ex: 34¢

Algorithm is "Greedy": One large coin better than two or more smaller ones

Cashier's algorithm. At each step, give the *largest* coin valued ≤ **the amount to be paid.**

Ex: $2.89

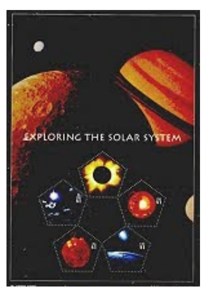
Coin-Changing: Does Greedy Always Work?

Observation. Greedy is sub-optimal for US *postal* denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

Counterexample. 140¢.
- Greedy: 100, 34, 1, 1, 1, 1, 1, 1.
- Optimal: 70, 70.

Algorithm is "Greedy", but also short-sighted – attractive choice now may lead to dead ends later.

Correctness is key!



Outline & Goals

"Greedy Algorithms"
- what they are

Pros
- intuitive
- often simple
- often fast

Cons
- often incorrect!

Proofs are crucial. 3 (of many) techniques:
- stay ahead
- structural
- exchange arguments

START WED JAN 23

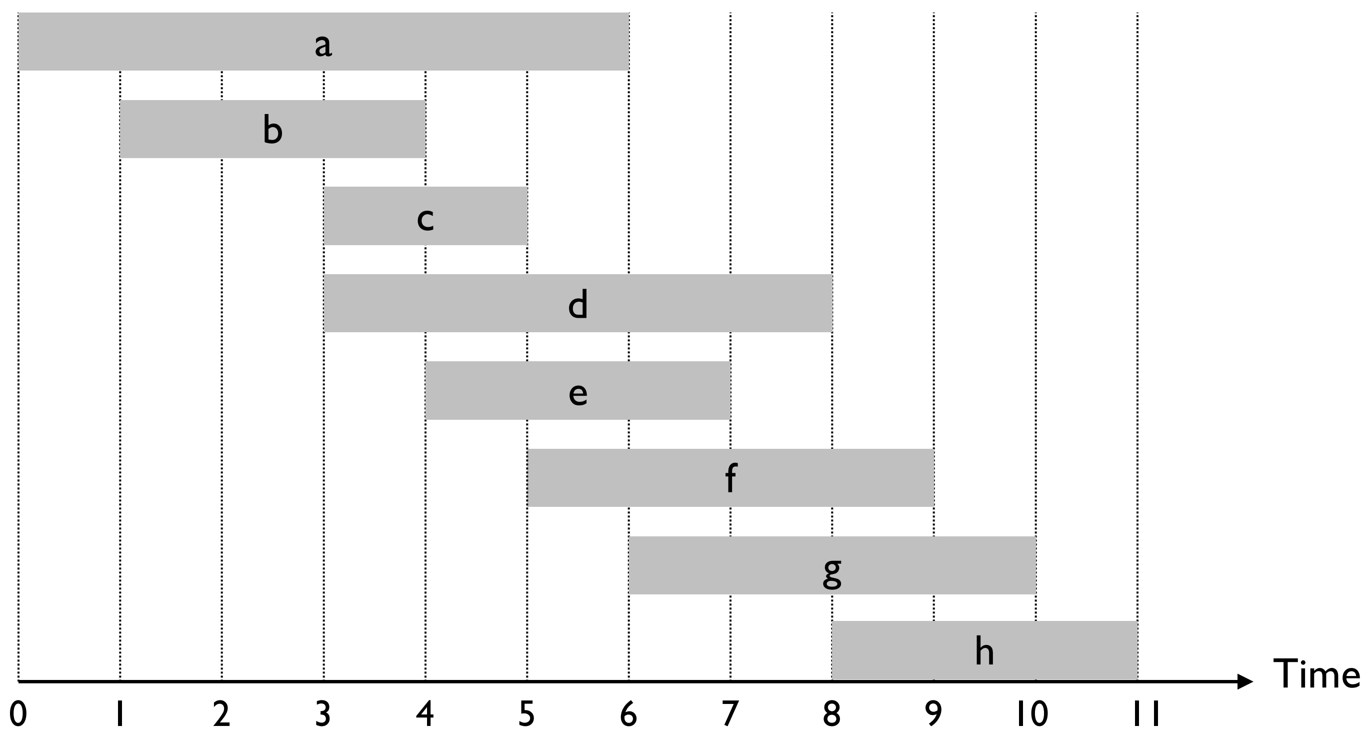
4.1 Interval Scheduling

Proof Technique 1: "greedy stays ahead"

argue that each step of this algorithm is even/better than other possible choices.

Interval scheduling.

- Job j starts at s_j and finishes at f_j.
- Two jobs compatible if they don't overlap.
- Goal: find max size subset of mutually compatible jobs.



Greedy template. Consider jobs in some order. Take next job provided it's compatible with the ones already taken.

- What order?
- Does that give best answer?
- Why or why not?
- Does it help to be greedy about order?

Interval Scheduling: Greedy Algorithms

Greedy template. Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

different order to select our jobs in…

[Earliest start time] Order jobs by ascending start time s_j

[Earliest finish time] Order jobs by ascending finish time f_j

[Shortest interval] Order jobs by ascending interval length $f_j - s_j$

[Longest Interval] Reverse of the above

[Fewest conflicts] For each job j, let c_j be the count the number of jobs in conflict with j. Order jobs by ascending c_j

Can You Find Counterexamples?

E.g., Longest Interval:

Others?:

END FRI JAN 18

Interval Scheduling: Greedy Algorithms

Greedy template. Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

breaks earliest start time

breaks shortest interval

breaks fewest conflicts

Greedy algorithm. Consider jobs in *increasing order of finish time*. Take each job provided it's compatible with the ones already taken.

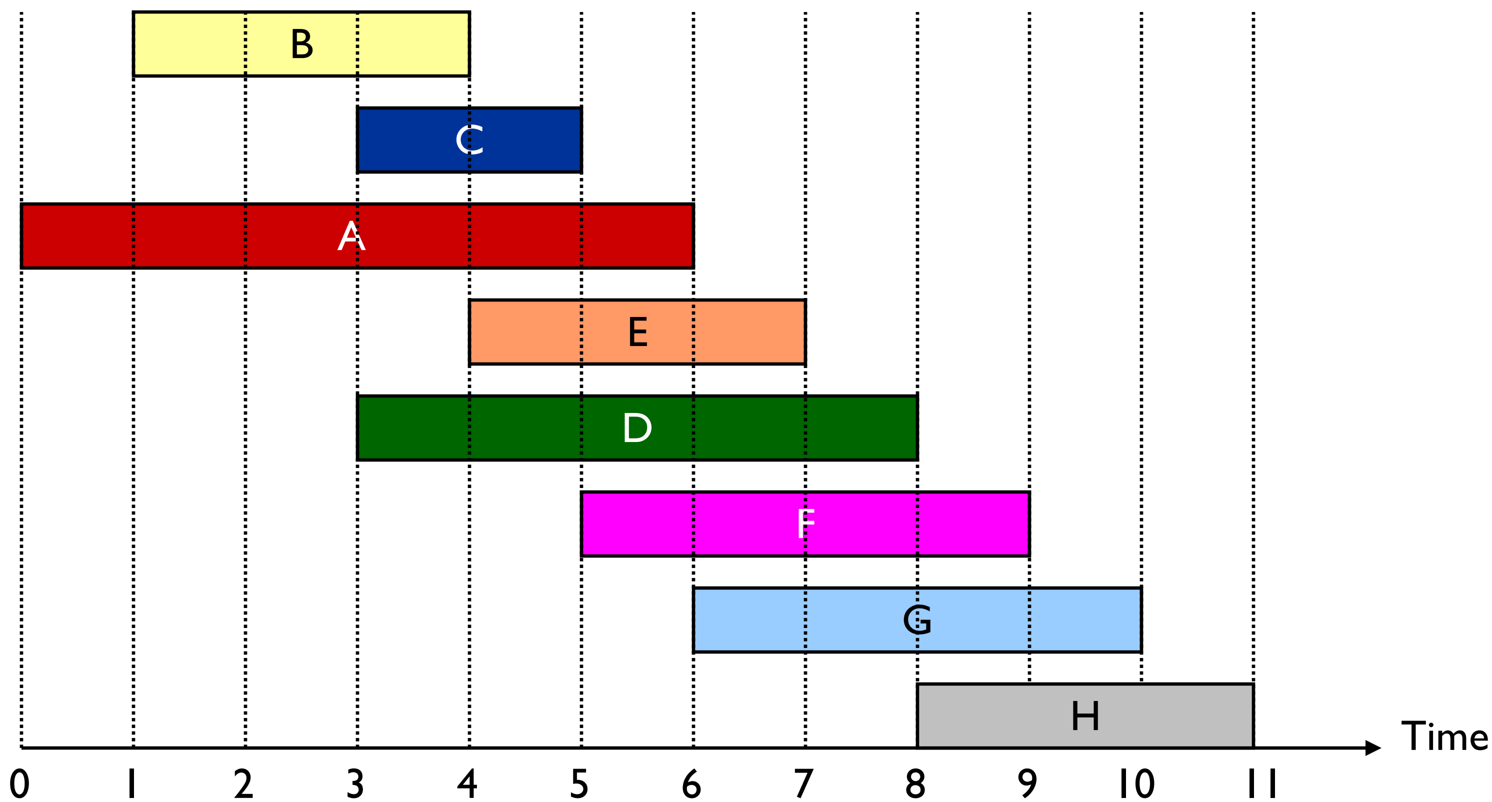
```
Sort jobs by finish times so that
f1 ≤ f2 ≤ ... ≤ fn.

  jobs selected
A ← φ
for j = 1 to n {
   if (job j compatible with A)
      A ← A ∪ {j}
}
return A
```

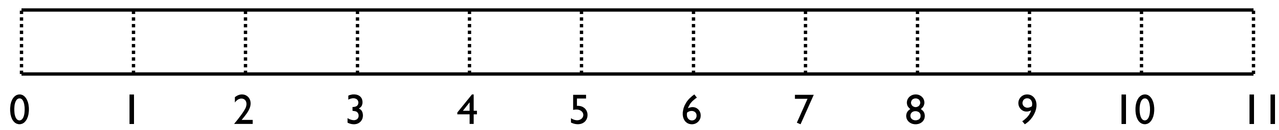
Does the start time of j come AFTER the finish time of the last interval I added to my collection A (no need to check if A overlaps others)

Implementation. O(n log n).

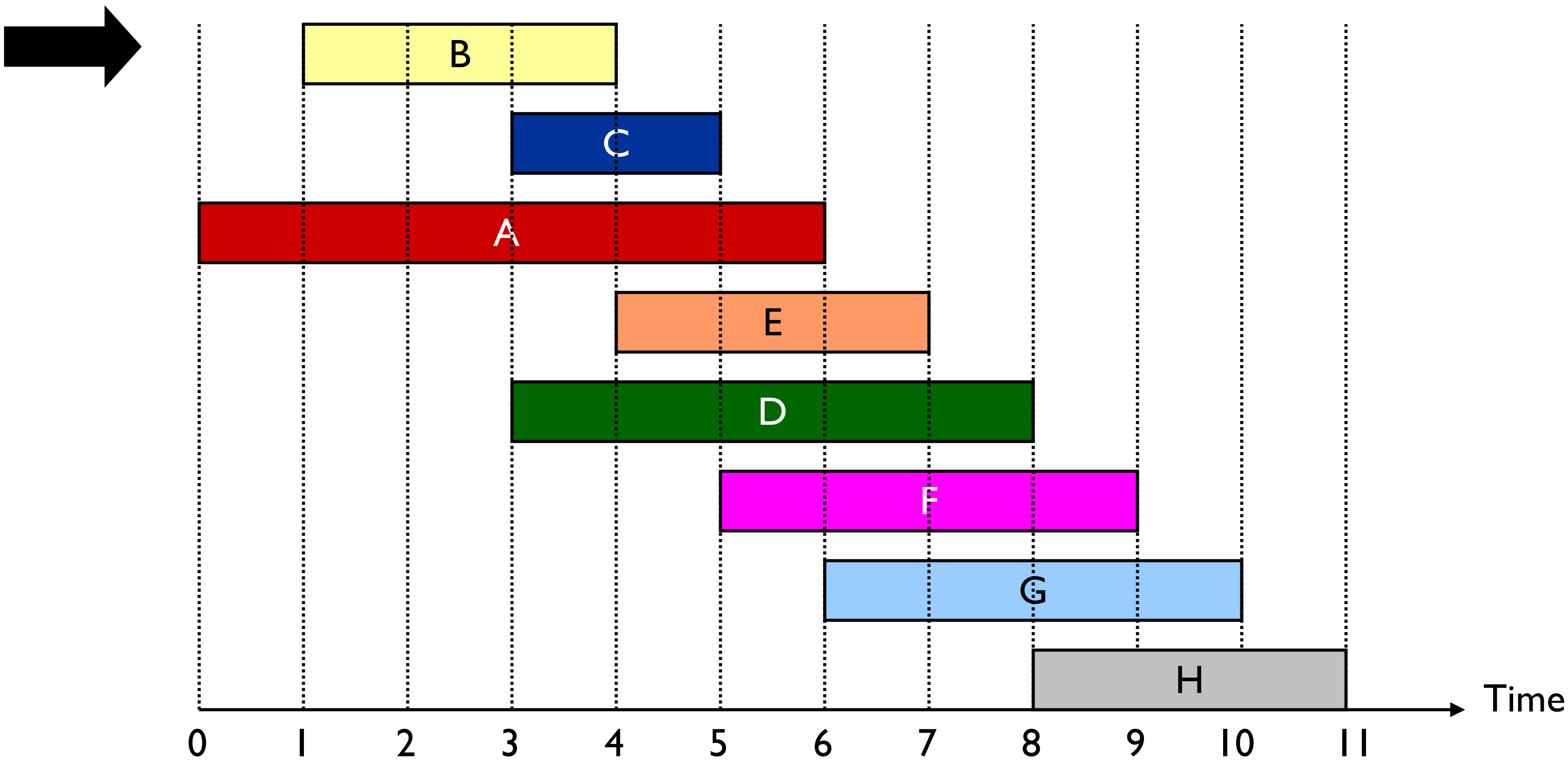
- Remember job j^* that was added last to A.
- Job j is compatible with A if $s_j \geq f_{j^*}$.

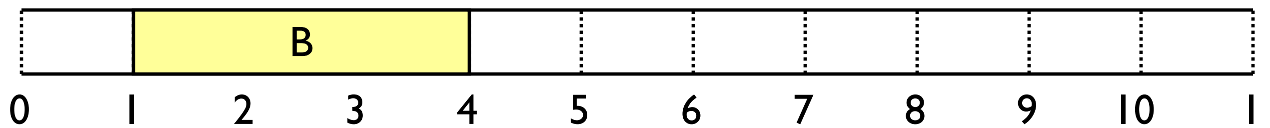
Interval Scheduling



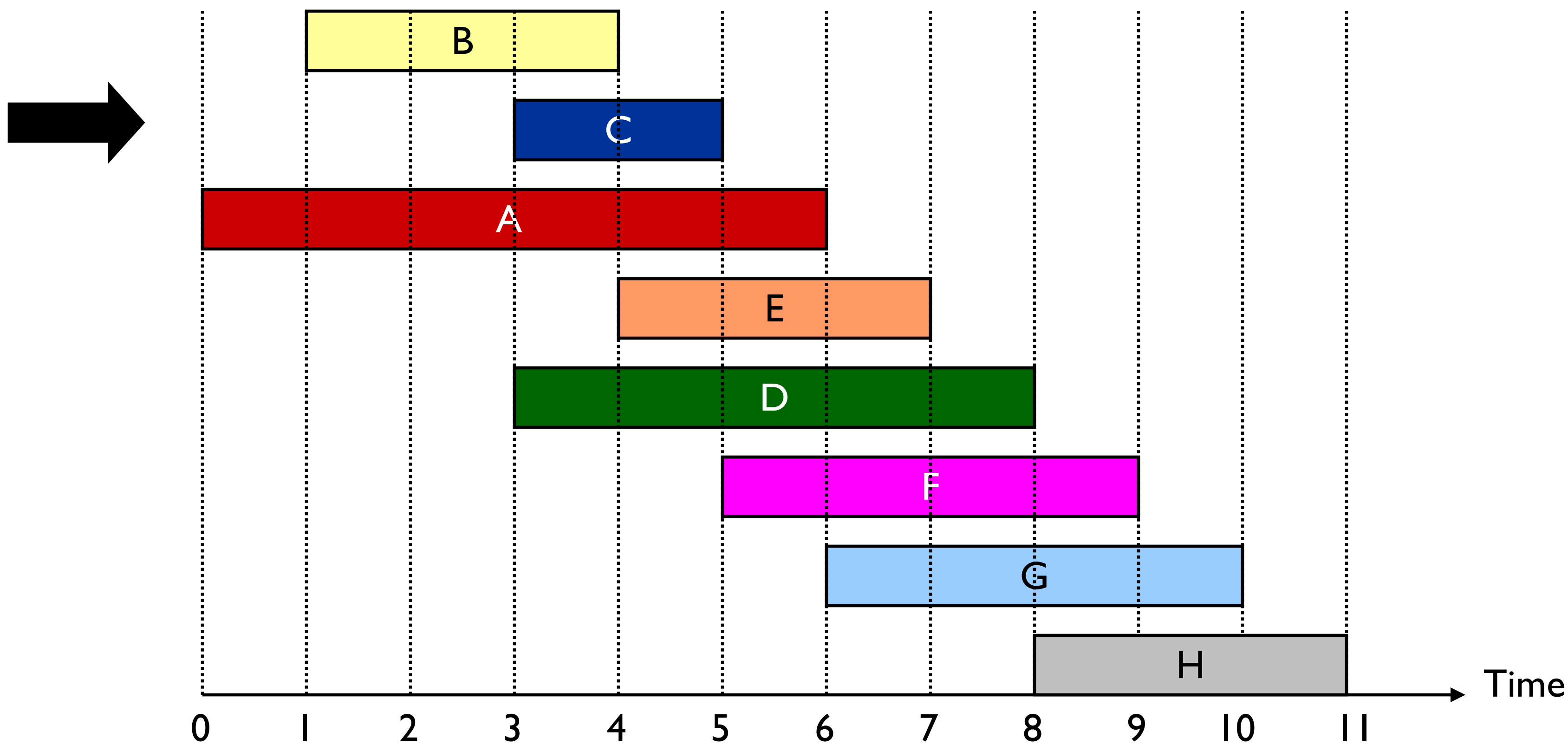


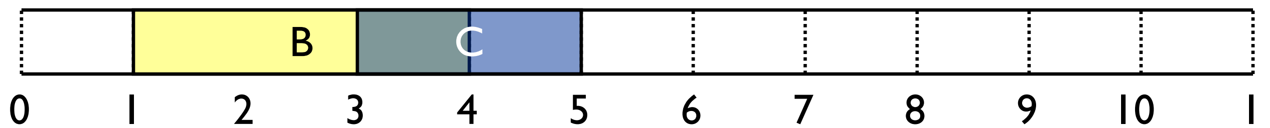
Interval Scheduling



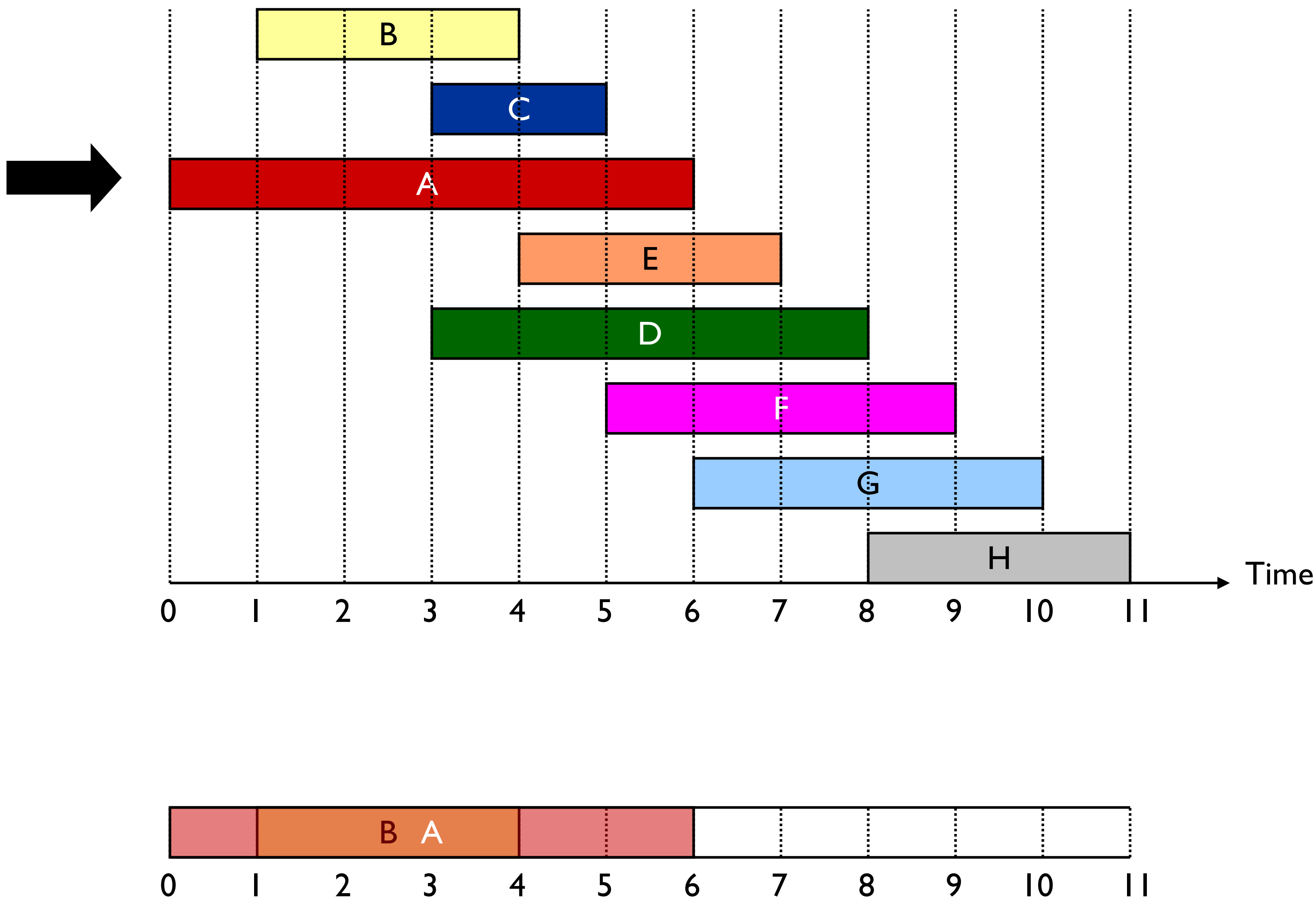


Interval Scheduling

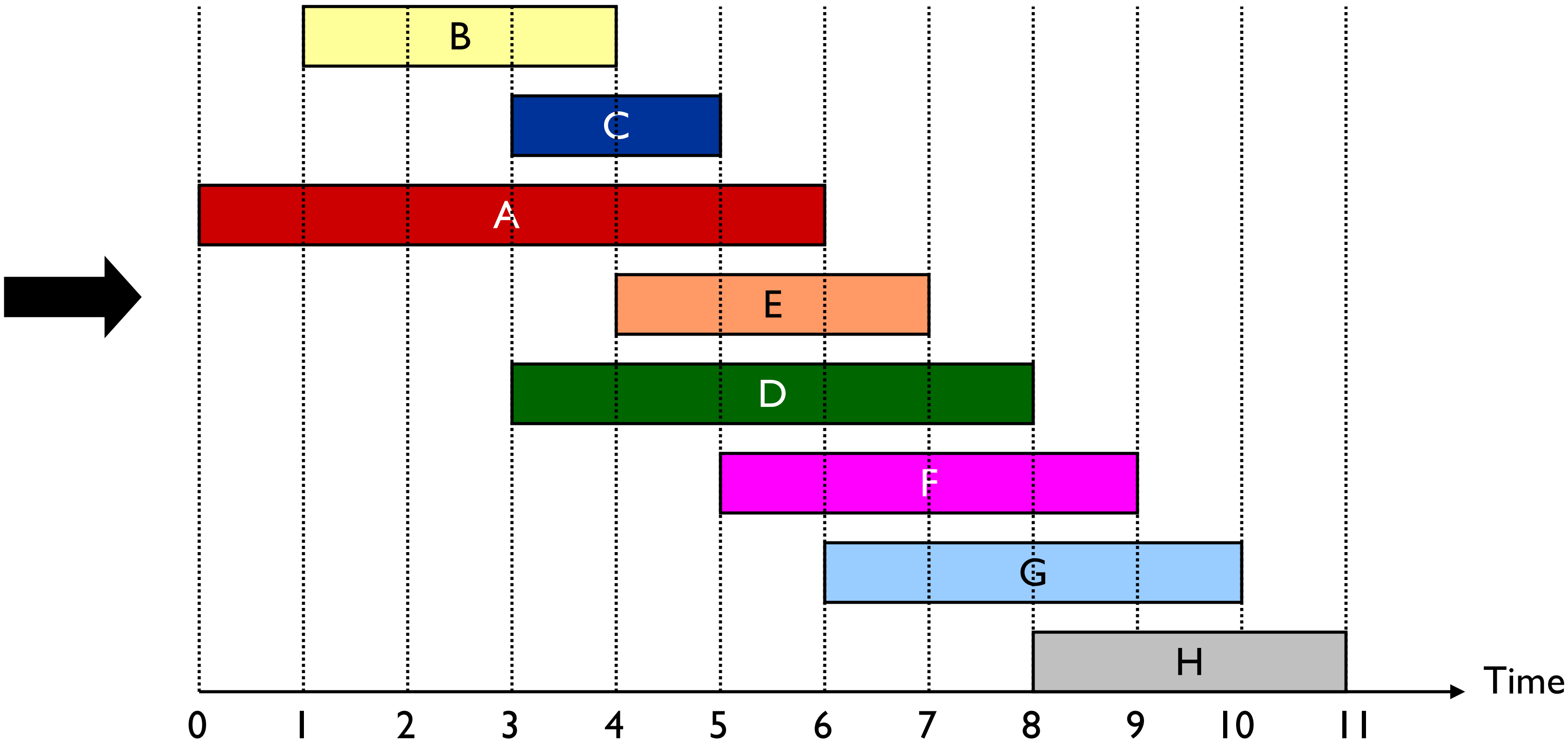


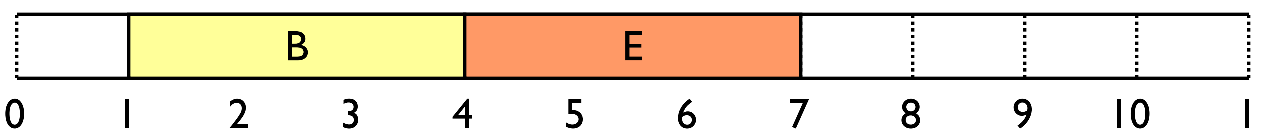


Interval Scheduling

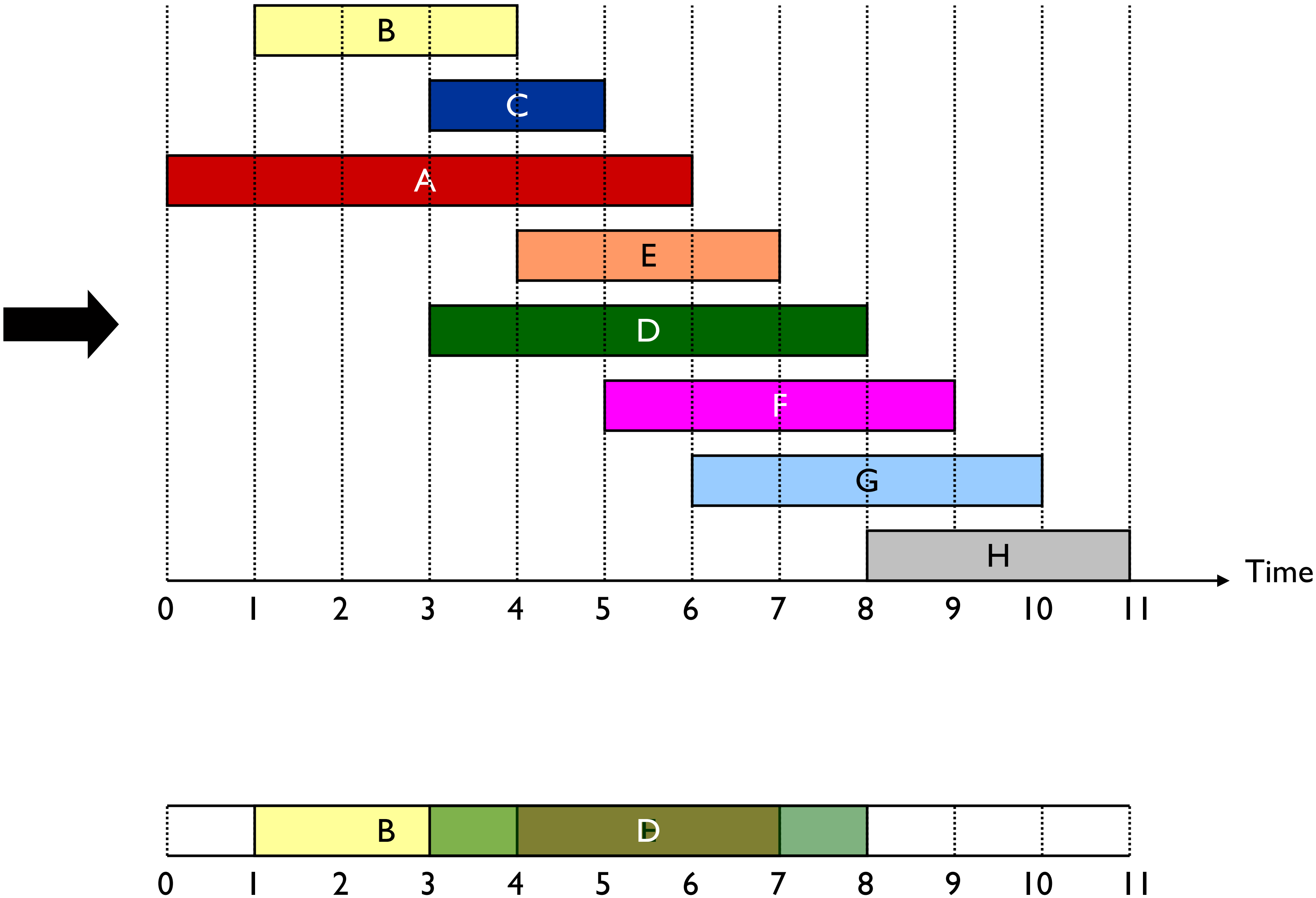


Interval Scheduling

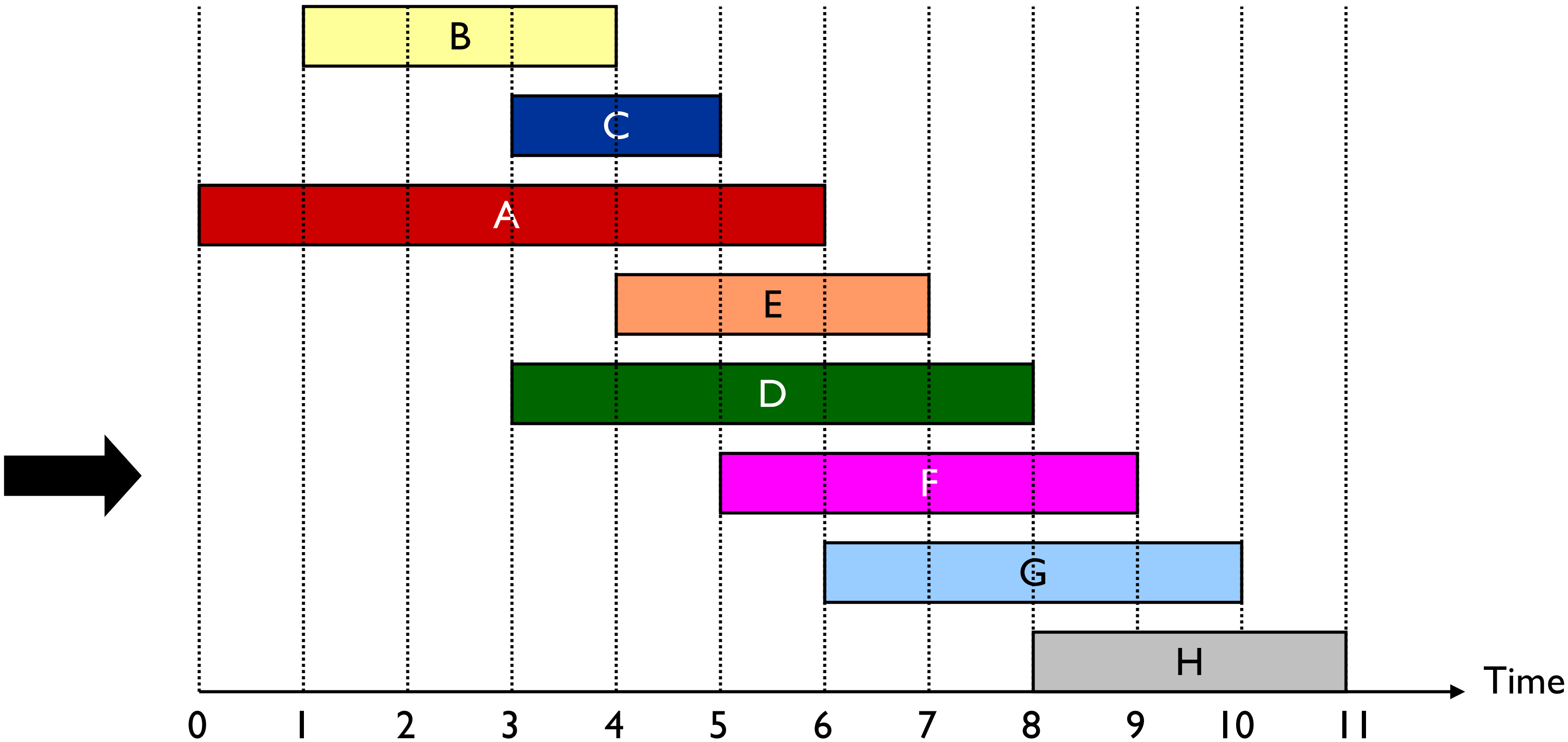


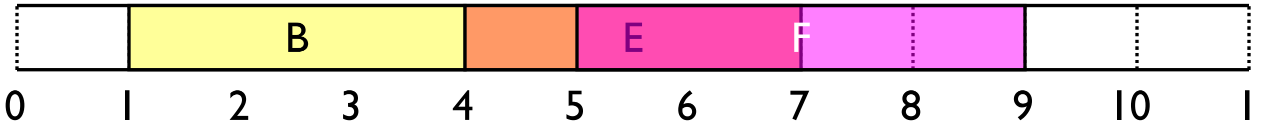


Interval Scheduling

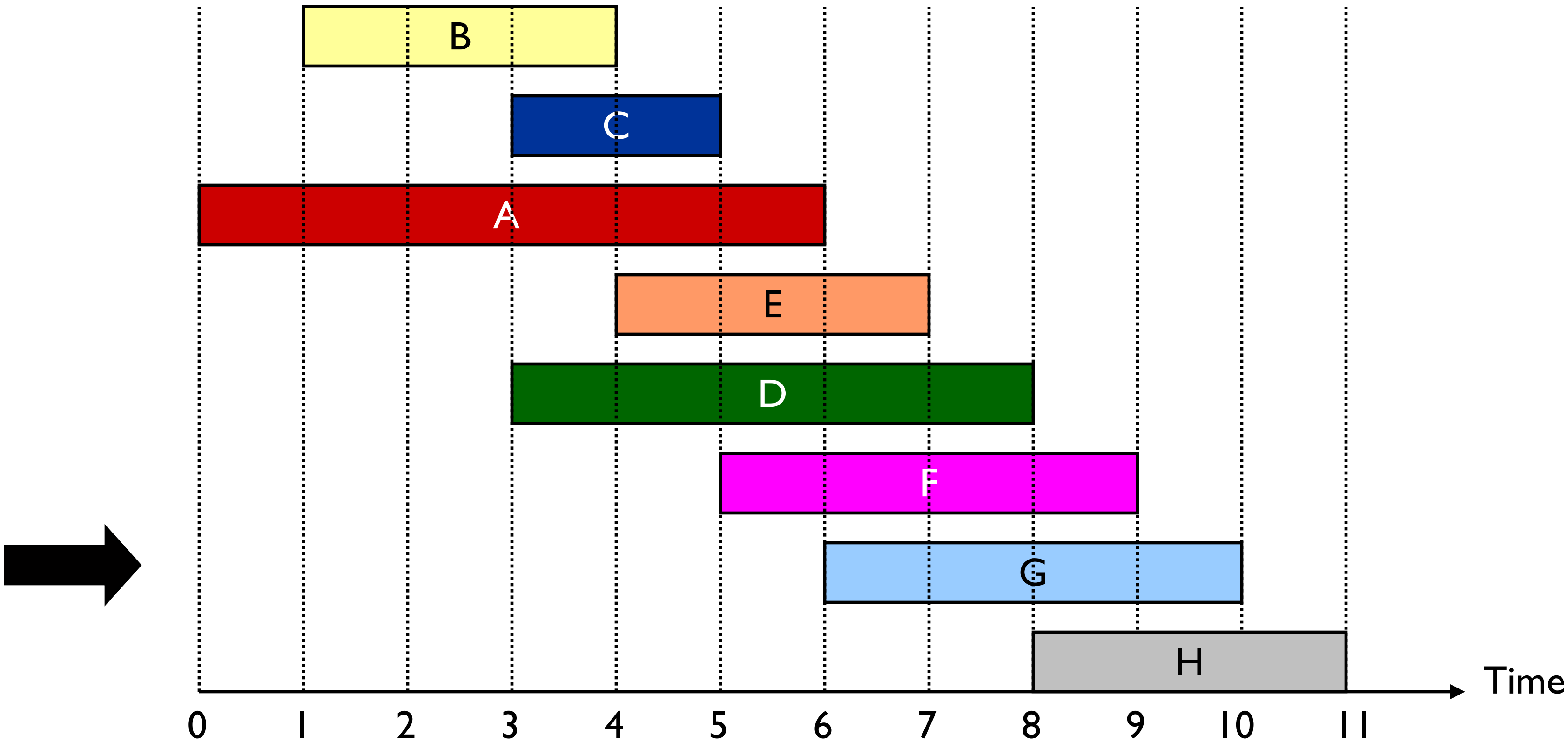


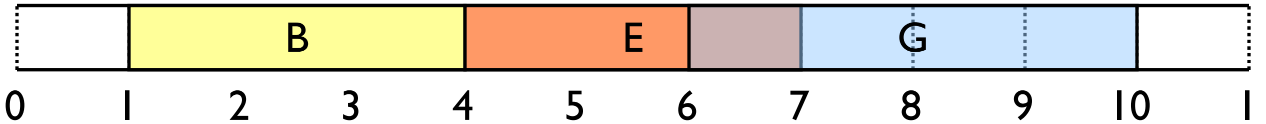
Interval Scheduling



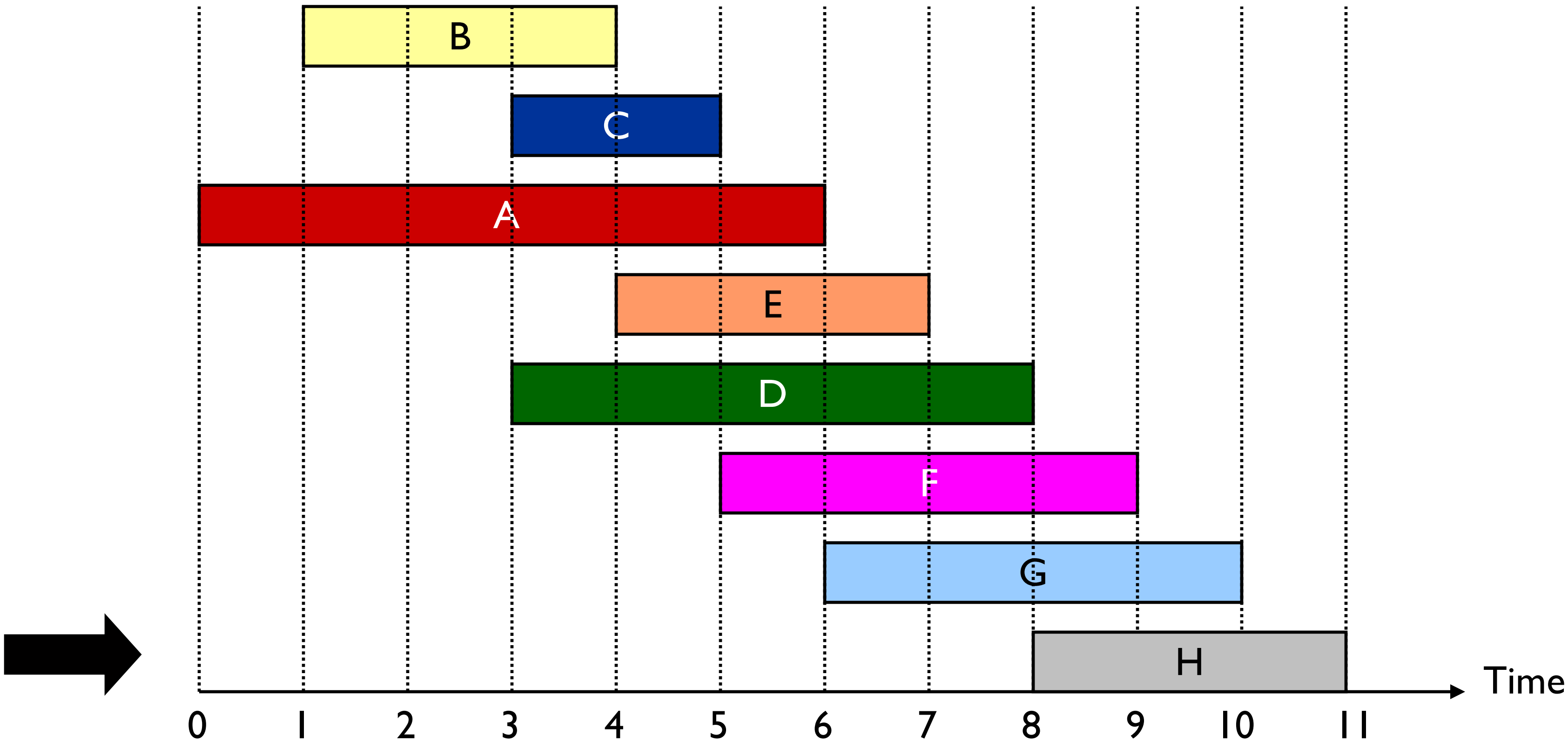


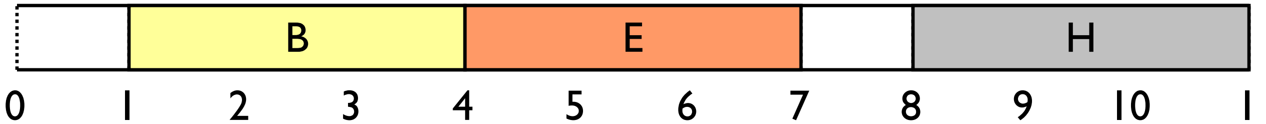
Interval Scheduling





Interval Scheduling





Interval Scheduling: Correctness

Theorem. *Earliest Finish First* Greedy algorithm is optimal.

COMPLICATION: Greedy algorithm produces an optimal solution, BUT it may not be the only one. Keep that in mind during your proof.

Pf. ("greedy stays ahead") i.e. picked by the algorithm

Let $g_1, \ldots g_k$ be greedy's job picks, $j_1, \ldots j_m$ those in some ~~optimal~~ solution

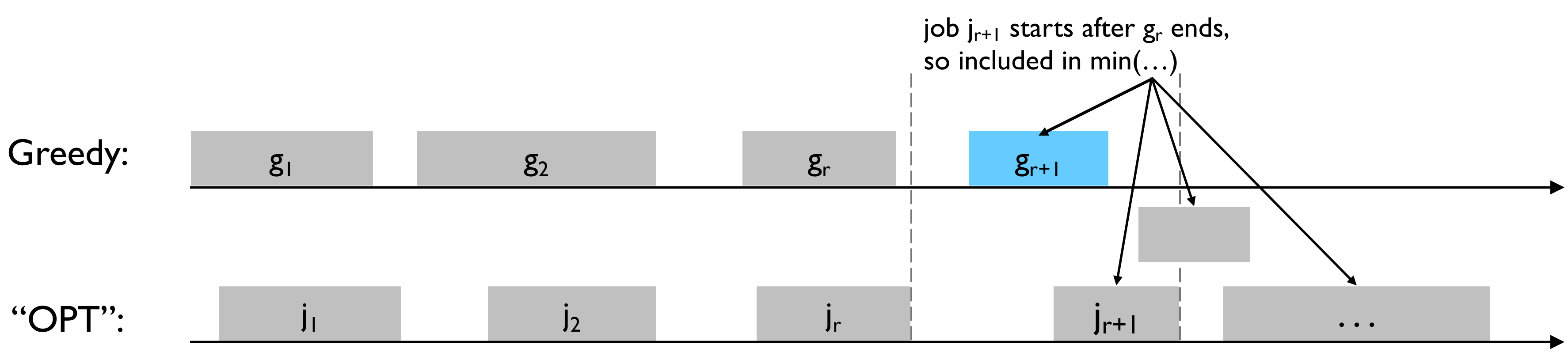
Show $f(g_r) \leq f(j_r)$ by induction on r.

algorithm specifies this

Basis: g_1 chosen to have min finish time, so $f(g_1) \leq f(j_1)$

Ind: $f(g_r) \leq f(j_r) \leq s(j_{r+1})$, so j_{r+1} is among the candidates considered by greedy when it picked g_{r+1}, & it picks min finish, so $f(g_{r+1}) \leq f(j_{r+1})$

Similarly, $k \geq m$, else j_{k+1} is among (nonempty) set of candidates for g_{k+1}



after I chose g1, since f(g1) <= f(j1), and as j2 is chosen next in optimal solution, it must not overlap j1. Thus, it must not overlap g1. Thus, j2 is in candidates considered for choosing g2. Thus, f(g2) <= f(j2) because our greedy algorithm chooses the min finish time.

Then show that we have at least as many greedy jobs g as optimal jobs j. If g ended at gk and j had another element jk+1, then as f(gk) <= f(jr), we could have picked jk+1 in our greedy algorithm