Greedy Algorithms

Week 5 Objectives

- Subproblem structure
- Greedy algorithm
- Mathematical induction application
- Greedy correctness

Subproblem Optimal Structure

- Divide and conquer - optimal subproblems
- divide PROBLEM into SUBPROBLEMS, solve SUBPROBLEMS
- combine results (conquer)
- critical/optimal structure: solution to the PROBLEM must include solutions to subproblems (or subproblem solutions must be combinable into the overall solution)
- PROBLEM = {DECISION/MERGING + SUBPROBLEMS}

Optimal Structure - GREEDY

- PROBLEM = {DECISION/MERGING + SUBPROBLEMS}
- GREEDY CHOICE: can make the DECISION without solving the SUBPROBLEMS
  - the GREEDY CHOICE looks good at the moment, and it is globally correct
  - example : pick the smallest value
  - solve SUBPROBLEMS after decision is made
- GREEDY CHOICE: after making the DECISION, very few SUBPROBLEMS to solve (typically one)

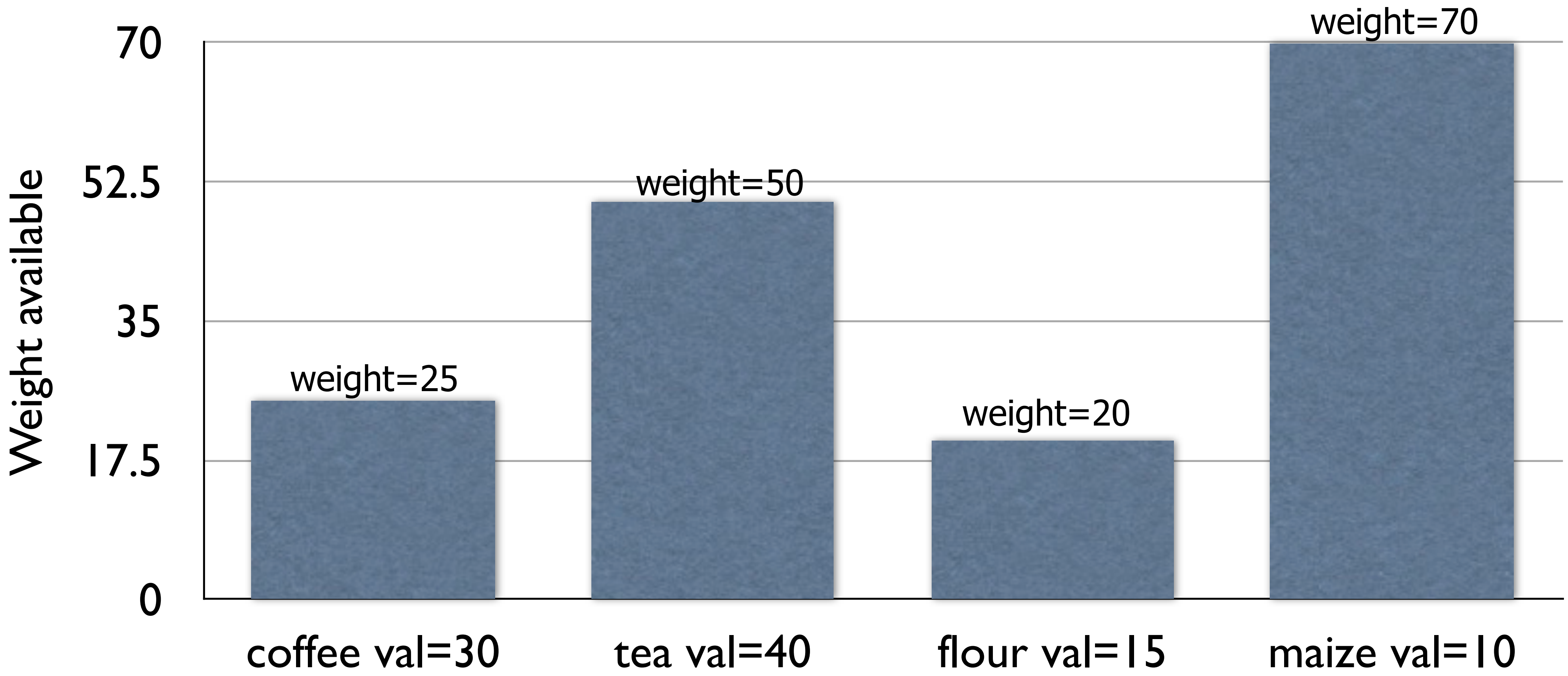
Optimal Structure - NON GREEDY

- Cannot make a choice decision/CHOICE without solving subproblems first
- Might have to solve many subproblems before deciding which results to merge.

Ex: Fractional Knapsack

- fractional goods (coffee, tea, flour, maize...) sold by weight
- supply (weights/quantities available) $w1,w2,w3,w4...$
- values (totals) $v1,v2,v3,v4...$
  - ex: coffee $w1$=10pounds; coffee overall value $v1$=$40
- knapsack capacity (weight) = W
- task : fill the knapsack to maximize value

Ex: Fractional Knapsack



- naive approaches may lead to a bad solution
  - choose by biggest value - tea first
  - choose by smallest quantity - flour first
- choose by quality is correct- coffee first
  - $q_{coffee}=30/25$; $q_{tea}=40/50$; $q_{flour}=15/20$; $q_{maize}=10/70$

Ex: Fractional Knapsack

- solution: compute item quality (value/weight)
- $q_i = v_i / w_i$
- sort items by quality $q1 > q2 > q3 > \ldots$
- LOOP
  - take as much as possible of the best quality
  - if knapsack full, STOP
  - if stock depletes (knapsack not full), move on to the next quality item, repeat
  - END LOOP

Fractional Knapsack - greedy proof

- proving now that the greedy choice is optimal
  - meaning that the solution includes the greedy choice.
- greedy choice: take as much as possible form best quality (below item with quality q1)
  - items available sorted by quality: q1>q2>q3>..., greedy choice is to take as much as possible of item 1, that is quantity w1
- contradiction/exchange argument
  - suppose that best solution doesnt include the greedy choice: SOL=(r1,r2,r3,...) quantities chosen of these items, and that r1 is not the max quantity available (of max quality item), r1<w1
  - create a new solution SOL' from SOL by taking more of item 1 and less of the others
    - e=min(r2,w1-r1); SOL'=(r1+e,r2-e,r3,r4...)
    - value(SOL') - value(SOL) = e(q1-q2)>0 which means SOL' is better than SOL: CONTRADICTING that SOL is best solution

Fractional Knapsack - greedy proof

- english explanation:
  - say coffee is the highest quality,
  - the greedy choice is to take max possible of coffee which is w1=10pounds
- contradiction/exchange argument
  - suppose that best solution doesnt include the greedy choice: SOL=(8pounds coffee, r2 of tea, r3 flours,...) r1=8pounds<w1=10pounds
  - create a new solution SOL' from SOL by taking out 2pounds of tea and adding 2 pounds of coffee; e=2pounds
    - e=min(r2,w1-r1); SOL'=(r1+e,r2-e,r3,r4...)
    - value(SOL') - value(SOL) = e(q1-q2)>0 which means SOL' is better than SOL: CONTRADICTING that SOL is best solution

Activity Selection Problem

- S=set of n activities given by start and finish time $a_i = (s_i, f_i)$ $i=1:n$, $f_i > s_i$
- Determine a selection that gives a maximal set
  - select maximum number of activities
  - no overlapping activities can be selected

Activity Selection Problem

- Greedy solution: sort activities by their finishing time
  - $f1<f2<f3\ldots$
  - select the activity that finishes first $a = (s_1, f_1)$
  - discard all overlapping activities with selected one : discard all activities with starting time $s_i<f_1$
  - repeat
- intuition: activity that finishes first is the one that leaves as much time as possible for other activities

Activity Selection Problem

- Proof of greedy choice optimality
  - activities sorted by finishing time f1<f2<f3...
  - greedy choice pick the activity a with earliest finishing time f1
  - want to show that activity a is included in one of the best solutions (could be more than one optimal selection of activities)
- Exchange argument
  - SOL a best solution.
  - if SOL includes a, done.
  - suppose the best solution does not select a, SOL= (b,c,d,...) sorted by finishing time $f_b<f_c<f_d$. Then create a new solution that replaces b with a SOL'=(a, c, d,...).
    - This solution SOL' is valid, a and c dont overlap: $s_c>f_b>f_a$
    - SOL' is as good as SOL (same number of activities) and includes a

Mathematical Induction

- property P(n) = {TRUE, FALSE} for n=integer
  - want to prove P(n)=TRUE for all n
- Base cases: P(n)=TRUE for any $n \leq n_0$
- Induction Step: prove P(n+1) for next value n+1
  - if P(t)=TRUE for certain values of t<n+1 then prove by mathematical derivation/arguments than P(n+1)=TRUE
- Then P(n) = TRUE for all n

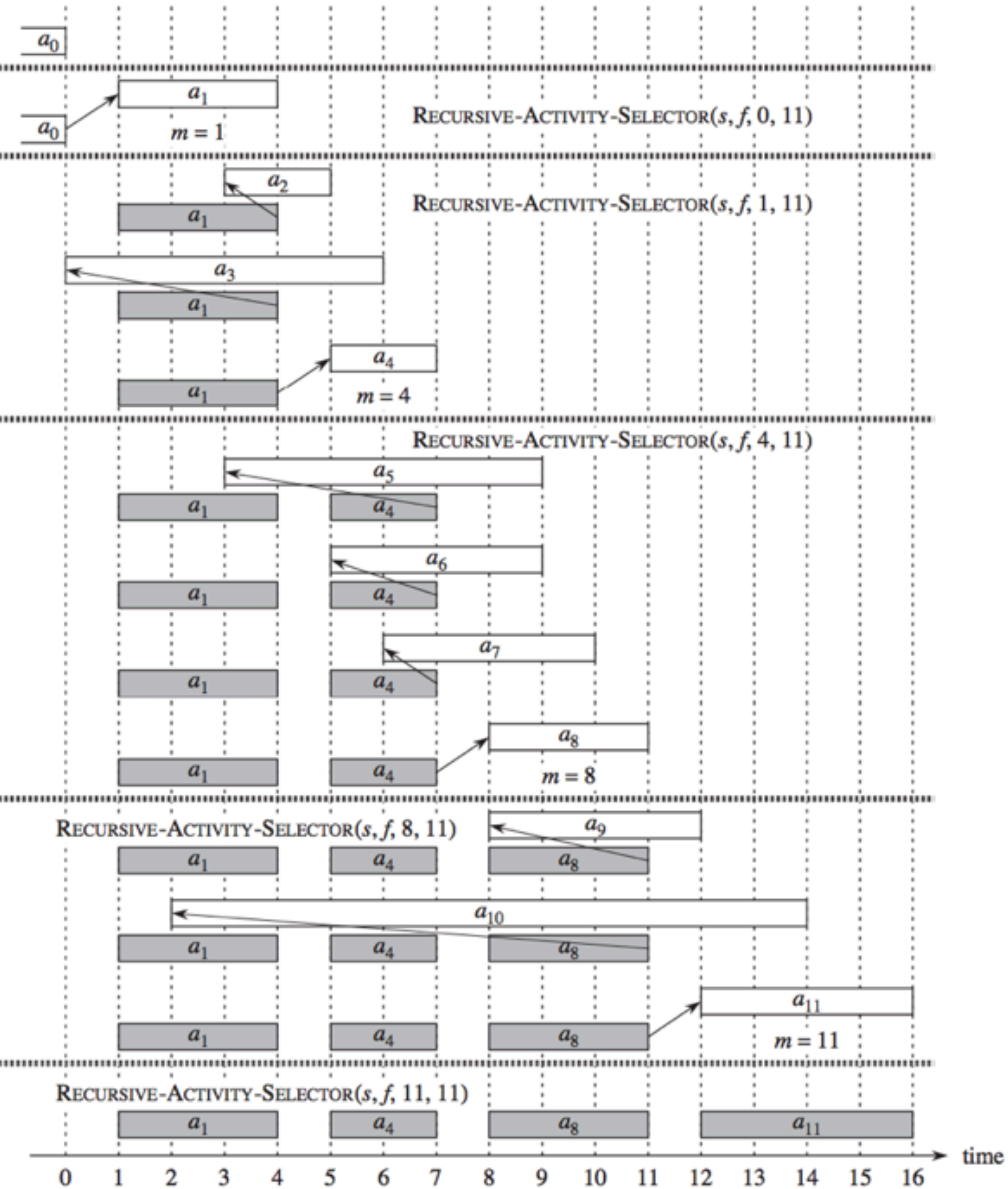
Mathematical Induction- Example

- $P(n)$: $1+2+3+...+n = n(n+1)/2$
- base case $n=1$: $1=1*2/2$ - correct
- induction step : lets prove $P(n+1)$ assuming $P(n)$
  - $P(n+1)$: $1+2+3+...+n + (n+1) = (n+1)(n+2)/2$.
  - assuming $P(n)$ TRUE : $1+2+3...+(n+1) = [1+2+3+...+n] + (n+1) = n(n+1)/2 + (n+1) = (n+1)(n+2)/2$; so $P(n+1)$ TRUE
- thus $P(n)$ TRUE for all $n>0$

Activity Selection – Induction Argument

- $s(a)$= start time; $f(a)$=finish time
- SOL=$\{a_1,a_2,\ldots,a_k\}$ greedy solution
  - chosen by earliest finishing time
- OPT = $\{b_1,b_2,\ldots,b_m\}$ optimal solution, sorted by finishing time; optimal means m max possible
- prove by induction that $f(a_i)\leqslant f(b_i)$ for all $i=1:k$
  - base case $f(a_1)\leqslant f(b_1)$ because $f(a_1)$ smallest in the whole set
  - inductive step: assume $f(a_{n-1})\leqslant f(b_{n-1})$. Then b_n is a valid choice for greedy at step n because $f(a_{n-1})\leqslant f(b_{n-1})\leqslant s(b_n)$. Since greedy picked a_n over b_n, it must be because an fits the greedy criteria $f(a_n)\leqslant f(b_n)$
- so $f(a_k)\leqslant f(b_k)$. If $m>k$ then any b_{k+1} item would also fit into greedy solution (CONTRADICTION) thus $m=k$

k	s_k	f_k
0	–	0
1	1	4
2	3	5
3	0	6
4	5	7
5	3	9
6	5	9
7	6	10
8	8	11
9	8	12
10	2	14
11	12	16




```
RECURSIVE-ACTIVITY-SELECTOR(s, f, k, n)
  m = k + 1
  while m ≤ n and s[m] < f[k]        // find
      m = m + 1
  if m ≤ n
      return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
  else return Ø
```