G	ire	ec	ly /	Alg	orit	hms	
					~ `		

Given a set of activities (s_i, f_i), we want to schedule the maximum number of non-overlapping activities.
 <u>GREEDY-ACTIVITY-SELECTOR</u> (s, f)

10/18/05

```
1. n = length[s]
```

```
2. S = \{a_1\}
```

3. i = 1

4. for m = 2 to n do

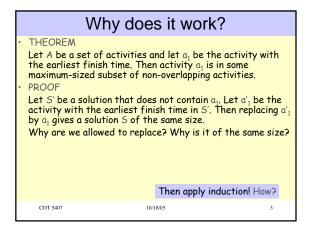
5. if s_m is not before f_i then 6. $S = S \cup \{a_m\}$

S = S U {a_m} i = m

7. 8. return S

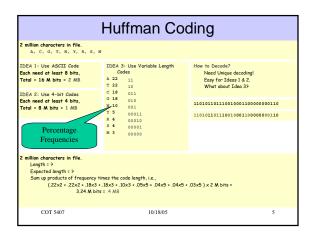
COT 5407

_	
	Example
•	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14] Sorted by finish times
•	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
٠	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
٠	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12] , [2,13], [12,14]
•	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12] , [2,13], [12,14]
•	[1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
	COT 5407 10/18/05 2

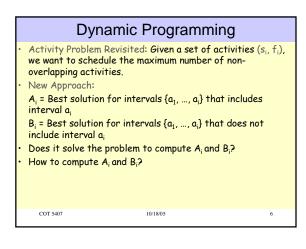


_				
	Greed	y Algorithm	s – Huffma	an Coding
•	contains 1,614, acids. There are of bits to store ~2.5 G bits How to improve	e 29.1 of 15-Feb- 107 sequence en 20 possible amin the compressed o or 300MB.	tries, comprising o acids. What is database?	Protein Database 505,947,503 amino the minimum number
•	Ala (A) 7.72 Arg (R) 5.24 Asn (N) 4.28 Asp (D) 5.28 Cys (C) 1.60 Idea: Use short codes for less for		Leu (L) 9.56 Lys (K) 5.96 Met (M) 2.36 Phe (F) 4.06 Pro (P) 4.87 e frequent amin	Ser (S) 6.98 Thr (T) 5.52 Trp (W) 1.18 Tyr (Y) 3.13 Val (V) 6.66 no acids and longer
	COT 5407	•)/18/05	4









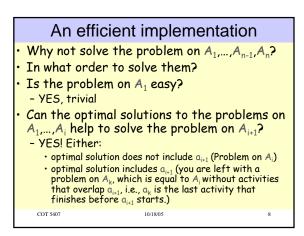
Dynamic Programming

- Activity Problem Revisited: Given a set of n activities $a_i = (s_i, f_i)$, we want to schedule the maximum number of non-overlapping activities.
- New Approach:

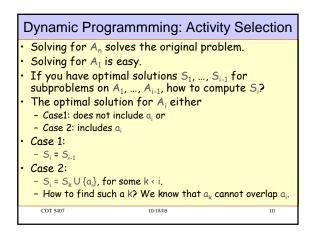
COT 5407

- Observation: To solve the problem on activities $A_n = \{a_1, \dots, a_n\}$, we notice that either
 - optimal solution does not include a_n (Problem on A_{n-1})
 - optimal solution includes a_n (Problem on A_k, which is equal to A_n without activities that overlap a_n, I.e., a_k is the last activity that finishes before a_n starts.)

10/18/05



Dynamic Programming: Activity Selection
Select the maximum number of non-overlapping activities from a set of n activities A = {a₁, ..., a_n} (sorted by finish times).
Identify "easier" subproblems to solve.
A₁ = {a₁}
A₂ = {a₁, a₂}
A₃ = {a₁, a₂, a₃}, ..., A_n = A
Subproblems: Select the max number of non-overlapping activities from A_i



Dynamic Programmming: A	Activity Selection
 <u>DP-ACTIVITY-SELECTOR</u> (s, f 1. n = length[s] 2.N[1] = 1 // number of activiti 3.F[1] = 1 // last activity in S₁ 4. for i = 2 to n do 5. let k be the last activity finishe 6. if (N[i-1] > N[k]) then // Case 7. N[i] = N[i-1] 8. F[i] = F[i-1] 	, es in S ₁ ed before s _i
9. else // Case 2 10. N[i] = N[k] + 1 11. F[i] = i	$\begin{array}{c} \text{How to output } S_n ? \\ & \text{Backtrack!} \\ \text{Time Complexity?} \\ & O(n \ lg \ n) \end{array}$
COT 5407 10/18/05	11

Dynamic Programming Features
 Identification of subproblems
 Recurrence relation for solution of subproblems
 Overlapping subproblems (sometimes)
 Identification of a hierarchy/ordering of subproblems
 Use of table to store solutions of subproblems (MEMOIZATION)
 Optimal Substructure

10/18/05

12

COT 5407

4