

329E: Algorithms for Bioinformatics

Tandy Warnow, Prof.

January 17, 2008

Some computational problems

1. Given a list of numbers, put it into sorted order
2. Given a map and a collection of cities, find the shortest tour that visits every city
3. Given a collection of people, find the largest subset of them that all know each other
4. Given a collection of people, determine if they can be put into 2 groups so that no two people in the same group know each other.
5. Given a collection of people, determine if they can be put into 3 groups, so that no two people in the same group know each other.

Running time

- You count the number of operations
 - Input/Output
 - Comparisons
 - Arithmetic
 - Assignments to variables
 - Etc.
- Relate this number to the size of the input

Finding the maximum in an array

- Input: array $A[1,2,3,\dots,n]$
- Output: index containing the maximum entry from array A , and the maximum entry
- Algorithm (in pseudocode):
- **Index**:=1
- **Max**:= $A[1]$
- For $i=2$ up to n DO:
 - If $A[i]>$ **Max**, then {set **Max**:= $A[i]$ and set **Index**:= i }
- Return **Index**, **Max**

Running time analysis

- **Index**:=1
- **Max**:=A[1]
- For i=2 up to n DO:
 - If A[i]>**Max**, then {set **Max**:=A[i] and set **Index**:=i}
- Return **Index**, **Max**

Running time: $n+3$ operations for the initialization of variables (array A, Index, max, and i), $n-1$ comparisons of array entries to Max, at most $3(n-1)$ changes of variables, and then one operation for returning Index and Max.

Total: $O(n)$.

Big-oh notation

- A function $f(n)$ is $O(g(n))$ if there is some pair of constants c and c' such that
- $f(n) \leq cg(n)$ whenever $n > c'$
- Examples:
 - $3n^2$ is $O(n^2)$
 - $87n^2+5000$ is $O(n^2)$
 - $5n^2$ is $O(n^3)$
 - But n^3 is *not* $O(n^2)$

Sorting

- Input: array $A[1,2,\dots,n]$
- Output: array $A'[1,2,\dots,n]$ with the same entries as A , but in sorted order (from lowest to highest)
- Algorithm: use the previous algorithm as a subroutine!

Algorithm

- Given array $A[1,2,\dots,n]$
- For $j=n$ down to 2 DO:
 - Find index of max entry in $A[1,2,\dots,j]$
 - Swap $A[\text{index}]$ with $A[j]$
- Running time: $O(n^2)$

Maximum clique problems

- A clique in a graph is a subset of the vertices so that every pair of vertices in the subset are adjacent
- A maximum clique is a clique of maximum size (number of vertices)

The decision problem for max clique

- Input: graph G and integer k
- Output: YES if G has a clique of k vertices, otherwise NO

The optimization problem for max clique

- Input: graph G
- Output: the largest k such that G has a clique of k vertices

The construction problem for max clique

- Input: graph G
- Output: a largest clique in G

Algorithms and Problems

- Problems come in various forms:
 - **Yes/no**
 - **Optimization**
 - **Construction**
- Examples:
 - Does the graph have a k -clique? **(Yes/No)**
 - What is the size of the max clique in the graph? **(Optimization)**
 - Find a max clique in the graph. **(Construction)**

Solving an optimization problem using an oracle for the decision problem

- Using an oracle for the existence of a clique of size k , to find the maximum size clique
- For $k=n$ (# vertices) down to 1 DO:
 - If G has a k -clique, return k
- Note: if the Yes/No problem can be answered in polynomial time, then so can the optimization problem

Optimization and Construction

- We can also find a max clique in a graph, using an “oracle” to answer the optimization problem!

Constructing a max clique, using an oracle

- Let K be the size of the max clique in the graph G
- List the vertices of G v_1, v_2, \dots, v_n
- For $i=1$ up to n , DO:
 - If $G'=G-\{v_i\}$ has a max clique of size K , replace G by G'
- Return the vertices of G' .

- Note: it takes only n calls to the oracle to construct the maximum clique. So if the optimization problem can be solved in polynomial time, so can the construction problem.

Problems and algorithms

- Problems come in various forms, but solving the yes/no problem allows you to solve everything else (optimization and construction)
- Algorithms are written in pseudocode
- Running time analyses just count basic operations, and are given in big-oh notation

How do we solve the decision problem for max clique?

- Input: graph G and integer k
- Output: YES if G has a clique of size k , NO otherwise

Algorithm design

- Brute force (will work)
- Greedy (sometimes works)
- Divide-and-conquer
- Dynamic programming
- Recursion

The Rock Game

- Two person game
- Starting condition: two piles of rocks
- Each person can do one of the following: remove one rock from each pile, or remove one rock from exactly one pile
- The person to remove the last rock wins

The Rock Game

- Yes/No problem: Given the number of rocks in each pile, determine if the first player wins
- Construction problem: For the same input, devise a winning strategy for the player who wins.