CS 173, Lecture B
Tandy Warnow

Topics for today

- Reminder about Examlet on Tuesday
- My office hours next week:
  - Tuesday 2-3 PM and Thursday 3-4 PM
- Office hours after next week will be based on response to poll sent via Piazza
- Basics of combinatorial counting

Examlet

- One problem about Dynamic Programming
- Several problems about big-oh
- One problem about counting

Examlet

- Give a Dynamic Programming algorithm for one of the following:
  - A two-person game
  - The longest increasing subsequence
  - Determine if you can make change for n cents using coins (e.g., 3-, 5-, and 7-cent coins)

Examlet

- Big-oh
  - Know the definition! (Exists constants…)
  - Be able to find the constants to establish that f is O(g)
  - Be able to quickly figure out whether one function is big-oh of another

Examlet

- Counting
  - One problem that will require you to be able to count something, using basic techniques
  - Know the basic techniques
  - Know the basic formulas
    - C(n,k)
    - P(n,k)
    - The number of functions from A to B

Using combinatorial counting

- Evaluating exhaustive search strategies:
  - Finding maximum clique
  - Determining if a graph has a 3-coloring
  - Finding a maximum matching in a graph
  - Determining if a graph has a Hamiltonial cycle or an Eulerian graph

Combinatorial counting

How many ways can you

- put n items in a row?
- pick k items out of n?
- pick subsets of a set of size n?
- assign k colors to the vertices of a graph?
- match up n boys and n girls?

Technique

- To count the number of objects, design an algorithm to generate the entire set of objects. Check if each object is created exactly once (if not, you will have to do a correction later).
- The algorithm's output can be seen as the leaves of a decision tree, and you can just count the leaves.

Putting n items in a row

Algorithm for generating all the possibilities:

- For i=1 up to n, DO
  - Pick an item from S to go in position i
  - Delete that item from the set S

Analysis: each way of completing this generates a different list.

The number of ways of performing this algorithm is **n!**

Number of subsets of a set of size n

Algorithm to generate the subsets of a set $S = \{s_1, s_2, s_3, \ldots, s_n\}$

- For i=1 up to n DO:
  - Decide if you will include s_i

Analysis: each subset is generated exactly once, and the number of ways to apply the algorithm is $2 \times 2 \times \ldots \times 2 = 2^n$.

k-coloring a graph

Let G have vertices $v_1, v_2, v_3, \ldots, v_n$

Algorithm to k-color the vertices:

- For i=1 up to n DO:
  - Pick a color for vertex v_i

Analysis: each coloring is produced exactly once, and there are k^n ways of applying the algorithm.

Matching n boys and girls

Algorithm:

- Let the boys be B_1, B_2, … B_n and let the girls be G_1, G_2, … G_n.
- For i=1 up to n DO
  - Pick a girl for boy B_i, and remove her from the set

Analysis: there are n ways to pick the first girl, n-1 ways to pick the second girl, etc., and each way produces a different matching.

Total: n!

Picking k items out of n

Algorithm for generating all the possibilities:

- For i=1 up to k, DO
  - Pick an item from S to include in set A
  - Delete that item from the set S

The number of ways of performing this algorithm is $n(n-1)(n-2)\ldots(n-k+1)=n!/(n-k)!$

But each set A can be generated in multiple ways - and we have overcounted!

Fixing the overcounting

Each set A of k elements is obtained through k! ways of running the algorithm. As an example, we can generate $\{s_1, s_5, s_3\}$ in 6 ways, depending upon the order in which we pick each of the three elements.

So the number of different sets is the number of ways of running the algorithm, divided by k!.

The solution is **$n!/[k!(n-k)!]$**

Summary (so far)

- To count the number of objects, design an algorithm to generate the entire set of objects. Check if each object is created exactly once (if not, you will have to do a correction later).
- The algorithm's output can be seen as the leaves of a decision tree, and you can just count the leaves.

Summary

- Number of orderings of n elements is n!
- Number of subsets of n elements is 2^n
- Number of k-subsets of n elements is n!/[k!(n-k)!]
- Number of k-colorings of a graph is k^n

More advanced counting

- What is the number of k-subsets of a set $S = \{s_1, s_2, s_3, \ldots, s_n\}$ that do not include s_1?
- What is the number of k-subsets of a set $S = \{s_1, s_2, s_3, \ldots, s_n\}$ that do include s_1?
- What is the number of orderings of the set S in which s_1 and s_2 are not adjacent?
- What is the number of orderings of the set S in which s_1 and s_2 are adjacent?

New techniques

- Count the complement
- Divide into disjoint cases, and count each case

Example

- What is the number of k-subsets of a set $S = \{s_1, s_2, s_3, \ldots , s_n\}$ that do not include s_1?
- Solution: same as number of k-subsets of $\{s_2, s_3, \ldots , s_n\}$. So $(n-1)!/[(n-1-k)!k!]$

Example

- What is the number of k-subsets of a set $S = \{s_1, s_2, s_3, \ldots, s_n\}$ that do include s_1?
- Solution: same as the number of (k-1)-subsets of $\{s_2, s_3, \ldots, s_n\}$, so

  $(n-1)!/[(n-k)!(k-1)!]$

Example

- What is the number of orderings of the set S in which s_1 and s_2 are adjacent?

- Solution: two cases:
  - Case 1) s_1 followed by s_2
  - Case 2) s_2 followed by s_1
- Same number of each type. Easy to see that there are (n-1)! of each type, so 2(n-1)! in total

Example

- Number of orderings of the set S in which s_1 and s_2 are not adjacent?
- This is the same as $n!-2(n-1)!$

Using combinatorial counting

- Evaluating exhaustive search strategies:
  - Finding maximum clique
  - Determining if a graph has a 3-coloring
  - Finding a maximum matching in a graph
  - Determining if a graph has a Hamiltonian cycle or an Eulerian tour