3137 Data Structures and Algorithms in C++

Lecture 10

Aug 9 2006

Shlomo Hershkop

1

Announcements

- Welcome to the Final Class!!!!
  - we will wrap up data structures and algorithms today
- you need to submit all work by this weekend!!!
- please fill out the course evaluation on courseworks
- Extra office hours next few days

2

Outline

- Graph
  - flow networks
  - connectivity
  - path exploration
- Algorithm designs
  - greedy
  - divide and conquer
  - random
  - heuristic
- Other DS
  - Game trees
  - General DS
- Final Review

3

reminder

- graph is just a collection of items and relationships
  - important for exploring theoretical and practical problems
  - representation
  - algorithms

4

Question

- Anyone hear about six degrees of separation ??
- Anyone know how many links you need to follow from one page to any other (using the most direct route) ??

5

Dijkstra

- can anyone tell me what dijkstra's algorithm is used for ??
- what is it ??

6

Problem: MST

- most of the time will have weight associated with the graph
- will want the minimum cost spanning tree connecting all nodes
- Example : laying down electric lines

7

solutions

- prim's algorithm
  - greedy solution
  - $O(V^2)$ but can be done in O(E log V) with heap
  - grows the tree one branch at a time
- kruskals algorithm
  - bottom up solution
  - O(E log V)
  - puts together forest of cheap ST and combine to get MST

8

prim's algorithm

- V is your list of nodes
- choose random node to start MST
- while V notEmpty()
  - choose cheapest edge outgoing from your MST
  - remove from V, add to MST
  - avoid cycles

9

kruskal's algorithm

- look at all edges
- for use the cheapest edge to connect two nodes
  - as long as no cycles created

10

flow networks

- sometimes we are interested in graph as a capacity problem:
  - Represent graph of capacity between two points
  - Want to see what is the max flow the graph can carry between the points
- application: oil pipeline, electricity grid, etc

11

- Given a graph with a flow capacity, what strategies can you think will allow you
  - any flow
  - max flow
  - min flow

12

max flow algorithm

- hand out from last class
- will be using extra graph for book keeping

13

undoing

- can augment the algorithm by adding back edges
- handout

14

Another Graph problem

- say you have a bunch of tasks which have to be done over time
- some can be done in parallel
- some have to wait for a specific task to be done before being able to be started
- think building construction, can't paint the inside without putting up the walls…etc

15

graph connectivity

- biconnected:
  - graph is biconnected is there are no vertices which can disconnect the graph
  - i.e no articulation points
    - points of failure
    - how to find them all??

16

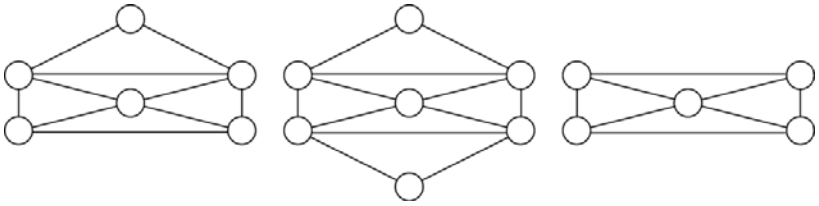
simple algorithm

- DFS
  - any vertex, DFS, number vertex as visited
  - check what is the lowest number you can reach from any edge and back edge
    - minimum of:
      - current number
      - lowest out edge
      - lowest of out edge

17

Euler Path

- can you have a path which starts and ends at the same edge and visits each edge exactly once ??



18

- solved in 1736

- key to solution:
  - even degrees
  - connected graph

  - best algorithm : O(E+V)

19

linear solutions

- so this is a linear solution for traveling to each edge

- what if you wanted to travel to each node

20

TSP

- Traveling salesman problem:
  - shortest route through n cities that visits each city once and returns to starting one
  - can be stated as finding the shortest Hamiltonian circuit of the graph
  - how many combinations of tours available ??

21

computing limitations

- at the beginning of the 1900's there was an open problem to show what would be the theoretical limitation of computing
- can a computer solve every problem

22

halting problem

- given a program and some input, determine if the program given this input ever stops

23

- generally the solution is undecideable
- but specific instances of this problem can be solved

24

poly time problems

- the algorithms we have studied are all solved in polynomial time

- O(1)
- O(n)
- O(n log n)
- $O(n^2)$
- $O(n^{10000})$

25

NP problems

- problems which can not be solved in poly time
- solutions can be checked in poly time

- so given a solution and graph, we can answer is it a Hamiltonian cycle in poly time

26

NP complete

- hardest of the NP problems
- they can all be mapped to each other

- example Map color ability
  - can you color a map so that no two countries which touch have the same color ??
  - can you use only 4 colors ??

27

- Algorithms
  - lets talk about the general approach of designing algorithms

28

Process

1. Understand the problem
2. Decide on exact vs approximate solution
3. Design your algorithm
4. Prove its correct
    1. might have to go back to 2 or 3
5. Analyze the algorithm
    1. might have to go back to 2 or 3
6. Code the algorithm

DON'T START WITH 6!!

29

understanding problem

- need to think about what you are trying to
    - represent
    - solve for
    - optimize
- think about computational requirements and resources
    - different if an experiment vs real world environment

30

Approximate solutions

- Why wouldn't you want an exact solution ??
  - can't be done
  - might be possible, very very slow
- might be part of an exact solution

31

Design issues

- Data structures can make or break your algorithm
  - sometimes can use the standard, more often than not cut-paste various models
- look for similar problems, how were they solved
- feel free to approach the problem in an unusual way
- documentation is essential
- try not to reinvent the wheel (Everywhere)

32

Proving

- check basic cases
- we've done very easy proofs, as mentioned some algorithms avg cases took years to find a proof
- what does it mean for approximate algorithm…how do you prove it works ?

33

Analyze it

- time resources
- space resources
- can it be simplified ?
- can you generalize it a little ?
- can it handle a broad set of inputs ?
- not always all steps necessary!

34

coding

- have lots of experience from this class ☺
- most of the time proof approximated by testing…remember to code this!!
  - packages available!
- choose language carefully
  - consider available language resources!
- don't loose efficiency of the algorithm with poor implementation

35

Classes of algorithms

- Lets talk about different types of algorithms
  - greedy
  - divide and conquer
  - dynamic algorithms
  - backtracking
  - decrease and conquer
  - randomized algorithms

36

Greedy Algorithm

- many examples that we have seen
- “grab what you can and run “ philosophy
- settle for an answer not quite best, hoping we get close

37

problem

- given a set of points in 2 dimensional space
- can you find closest pair of points to any point ?
- hmmm sounds familiar ??

38

- strategies ??
- runtimes ??
- improvements ?

39

- mentioned finding the median number in linear time
- any ideas on this ??

40

divide and conquer

- idea: to hard to solve whole problem so can solve bits and combine the results
- mergesorts

41

quick question

- Quick sort has a worst case behavior of ??
- so how do we solve it

42

median selection

- I mentioned that there is a linear algorithm for choosing the median
- anyone read it up ?

43

basic idea:

- divide your elements into groups of small size say 5
- find the median in each group…running time ???
- so we have N/2 medians
- find the medians of these medians
- how close are we to the median ??
- what does it do to quicksort ?

44

Dynamic algorithms

- divide the problem into overlapping sub problems
- the final solution will have repeated parts, so lets memoize them so only need to recompute them once
- fibonacci sequence

45

backtracking

- build up solution slowly, and when hit a dead end, back up and try another solution
  - uses the idea of promising paths and non-promising paths

46

problem

- how to place n queens on a n by n board
- lets do 4 queens

47

decrease and conquer

- idea: problem of size n can be solved as problem size n-1 + the i'th item
- think of some of the sorting routines

48

random algorithm

- why would you want to base your algorithm on random decisions ?

49

randomness

- for some hard problems
  - can not solve directly
  - we can get the correct answer with high probability
  - run efficiently in expectation!
- Example
  - computer networks….anyone use an ethernet network
  - anyone know how they work
  - how the tcp/ip protocol works ?

50

Problem

- given a Graph (undirected) we want to find the minimum number of edge deletions so that we can divide the graph into two distinct non-empty sets A and B
- real life example: say you have a complex network, what would be the minimum number of damages to it to split the network
- any ideas on this ?

51

contraction algorithm

- proposed in 92
- here is a sketch of the idea

52

- Review for final

53

basics

- should remember your fundamental data structures
  - lists
  - queues
  - stacks
- form the basis for advanced structures
  - trees
  - hash tables
  - graphs

54

implementation and run time

- You are familiar with runtime concepts
- implementation can make break a data structure
  - arrays, linked lists affect on implementation
- coding ideas we've covered
  - recursion
  - code layout
  - efficiency concerns

55

hash tables

- idea of hash table
- hash functions
- collision resolution
- scalability issues
- bloom filter hashing
- in use with other DS

56

Trees

- implementation details
- BST
- Traversals
- AVL
- B+ trees

- how are trees related to graphs ??

57

- sorting
  - Basic sorts
    - bubble
    - selection
    - insertion
  - Better
    - heapsort
  - Even Better
    - mergesort
    - quicksort
- analysis of sorts

58

sets

- idea of sets
- union / find
- path compression
- union by rank

59

Graphs

- implementation
- sorting
- shortest paths
- network flows
- MST
- Expansion
  - DFS
  - BFS
- Algorithms

60

Algorithms

- analyzing general run times
- comparing classes of algorithms
- getting a feel if it's a good algorithm

61

- Lessons Learned
  - Think then code!
- Try not to reinvent wheel or do things in n^3 or n! land ☺

62

practice I

- say you are standing in front of a wall which stretches infinitely to the left and right.
- there is a door somewhere N steps away, but you don't know which directions
- give an algorithm to find it

- now give a linear algorithm and exact runtime!

63

Practice II

- for prim's algorithm…if you have a choice between two edges, can you prove it makes no difference which one is chosen

64

Practice III

- You are given an array that contains N numbers.
- We want to determine if there are two numbers whose sum equals a given number K.
- for instance if the input is 8,4,1,6 and K=10, the answer is yes (use 4 and 6).
- A number may be used twice.
- Do the following:
  - Give an $O(N^2)$ algorithm to solve this problem
  - Give an O(N log N) algorithm to solve this problem.

65

Practice IV

- is it possible to implement insertion sort for sorting linked lists ?
- will it have the same $o(n^2)$ running as an array version ??

66

conclusion

- hope you have fun learning about the theory and programming of DS and Algorithms
- please remember its open your notes and book but nothing else (no random websites)
- please remember to fill out the online survey

67

where to next

- so what can you do now ☺

68

- Thank you
- good luck on the final….

69