Storage allocation issues

Prof. Ramin Zabih

http://cs100r.cs.cornell.edu

Cornell University
Computer Science

Administrivia

- Assignment 3 is out, due Friday
- Prelim on 10/11
  - Quiz 4 this Tuesday, 10/2
- Wednesday section will be led by RDZ
  - Topic: prelim review
- Wednesday evening 10/3, RPC205:
  - 7PM: Bobby Kleinberg on graphs (optional!)
- Gurmeet will do additional review in section 10/10

Memory allocation

- So far we just assumed that the hardware supplied us with a huge array M
  - When we need more storage, we just grab locations at the end
    - Keep track of next free memory location
  - What can go wrong?
    - Consider repeatedly adding, deleting an item
- Notice that when we delete items from a linked list we simply changed pointers so that the items were inaccessible
  - But, they still waste space!

Storage reclamation

- Someone has to figure out that certain locations can be re-used ("garbage")
  - If this is too conservative, your program will run slower and slower ("memory leak")
  - If it's too aggressive, your program will crash ("blue screen of death")
- Suppose your linked list was corrupted
  - ♦Why do computers crash when we read/write an arbitrary location? Must they??

Two basic options

- Computer keeps track of free storage
- Manual storage reclamation
  - Programmer has to explicitly free storage
  - Languages: C, C++, assembler
- Automatic storage reclamation
  - Computer will free storage for you
  - Languages: Matlab, Java, C#, Scheme, SML
- Basic tradeoff is program speed versus programmer effort
  - The computer isn't as smart as a programmer

Allocation issues

- Computer keeps a linked list of free storage blocks ("freelist")
  - For each block, keep size and location
  - When asked for new storage, get from freelist
- Surprisingly important question:
  - Which block do you supply?
- Different sized blocks
  - Actually, several different lists (for efficiency)
    - "Buddy block" system, see CS316/CS414 or http://www.memorymanagement.org/

Manual storage reclamation

- Programmers always ask for a block of memory of a certain size
  - In C, explicitly declare when it is free
- Desirable but complex invariants:
  - Everything should be freed when it is no longer going to be used
  - And, it should be freed exactly once!
  - Also want to minimize fragmentation
- Any serious C programmer writes their own memory allocation system!