CSE 373

MAY 31ST – EVAN'S FUN LECTURE

ASSORTED MINUTIAE

- Exam Review – Friday 4:30 – 6:00 EEB 105

ASSORTED MINUTIAE

- Exam Review – Friday 4:30 – 6:00 EEB 105
- Section will be exam review

ASSORTED MINUTIAE

- Exam Review – Friday 4:30 – 6:00 EEB 105
- Section will be exam review
- Friday will be an exam review

ASSORTED MINUTIAE

- **Exam Review – Friday 4:30 – 6:00 EEB 105**
- **Section will be exam review**
- **Friday will be an exam review**
- **Practice exams will be out tonight**
  - I will link some other practice finals as well, but they can be found on the 17wi page right now.

ASSORTED MINUTIAE

- **Exam Review – Friday 4:30 – 6:00 EEB 105**
- **Section will be exam review**
- **Friday will be an exam review**
- **Practice exams will be out tonight**
  - I will link some other practice finals as well, but they can be found on the 17wi page right now.
- **Exam: Tue Jun 6, 2:30 – 4:20**

TODAY'S LECTURE

- Interesting topics for implementation

TODAY'S LECTURE

- **Interesting topics for implementation**
  - Randomization Rant

TODAY'S LECTURE

- **Interesting topics for implementation**
  - Randomization Rant
  - Hardware constraints

RANDOMIZATION

- **Guess and check**

RANDOMIZATION

- **Guess and check**
  - How bad is it?

RANDOMIZATION

- **Guess and check**
  - How bad is it?
    - Necessary for some hard problems

RANDOMIZATION

- **Guess and check**
  - How bad is it?
    - Necessary for some hard problems
    - Still can be useful for some easier problems

RANDOMIZATION

- **If an algorithm has a chance P of returning the correct answer to an NP-complete problem in $O(n^k)$ time**

RANDOMIZATION

- **If an algorithm has a chance P of returning the correct answer to an NP-complete problem in $O(n^k)$ time**
  - P is our success probability

RANDOMIZATION

- **If an algorithm has a chance P of returning the correct answer to an NP-complete problem in $O(n^k)$ time**
  - P is our success probability
  - NP-complete means we can check a solution in $O(n^k)$ time, but we can find the exact solution in $O(k^n)$ time – very bad

RANDOMIZATION

- **If an algorithm has a chance P of returning the correct answer to an NP-complete problem in $O(n^k)$ time**
  - P is our success probability
  - NP-complete means we can check a solution in $O(n^k)$ time, but we can find the exact solution in $O(k^n)$ time – very bad
  - Suppose we want to have a confidence equal to α, how do we get this?

RANDOMIZATION

- **Even if P is low, we can increase our chance of finding the correct solution by running our randomized estimator multiple times**

RANDOMIZATION

- **Even if P is low, we can increase our chance of finding the correct solution by running our randomized estimator multiple times**
  - We can verify solutions in polynomial time, so we can just guess-and-check.

RANDOMIZATION

- **Even if P is low, we can increase our chance of finding the correct solution by running our randomized estimator multiple times**
  - We can verify solutions in polynomial time, so we can just guess-and-check.
  - How many times do we need to run our algorithm to be sure our chance of error is less than α?

RANDOMIZATION

- **Even if P is low, we can increase our chance of finding the correct solution by running our randomized estimator multiple times**
  - We can verify solutions in polynomial time, so we can just guess-and-check.
  - How many times do we need to run our algorithm to be sure our chance of error is less than α?

RANDOMIZATION

$$(1-p)^k = \alpha$$

RANDOMIZATION

$(1-p)^k = \alpha$

$k * \ln(1-p) = \ln \alpha$

$$k = \frac{(\ln \alpha)}{(\ln(1-p))}$$

$k = \log_{(1-p)} \alpha$

RANDOMIZATION

- Cool, I guess… but what does this mean?

RANDOMIZATION

- Cool, I guess… but what does this mean?
- Suppose P = 0.5 (we only have a 50% chance of success on any given run) and $\alpha = 0.001$, we only tolerate a 0.1% error

RANDOMIZATION

- Cool, I guess… but what does this mean?
- Suppose P = 0.5 (we only have a 50% chance of success on any given run) and $\alpha = 0.001$, we only tolerate a 0.1% error
- How many runs do we need to get this level of confidence?

RANDOMIZATION

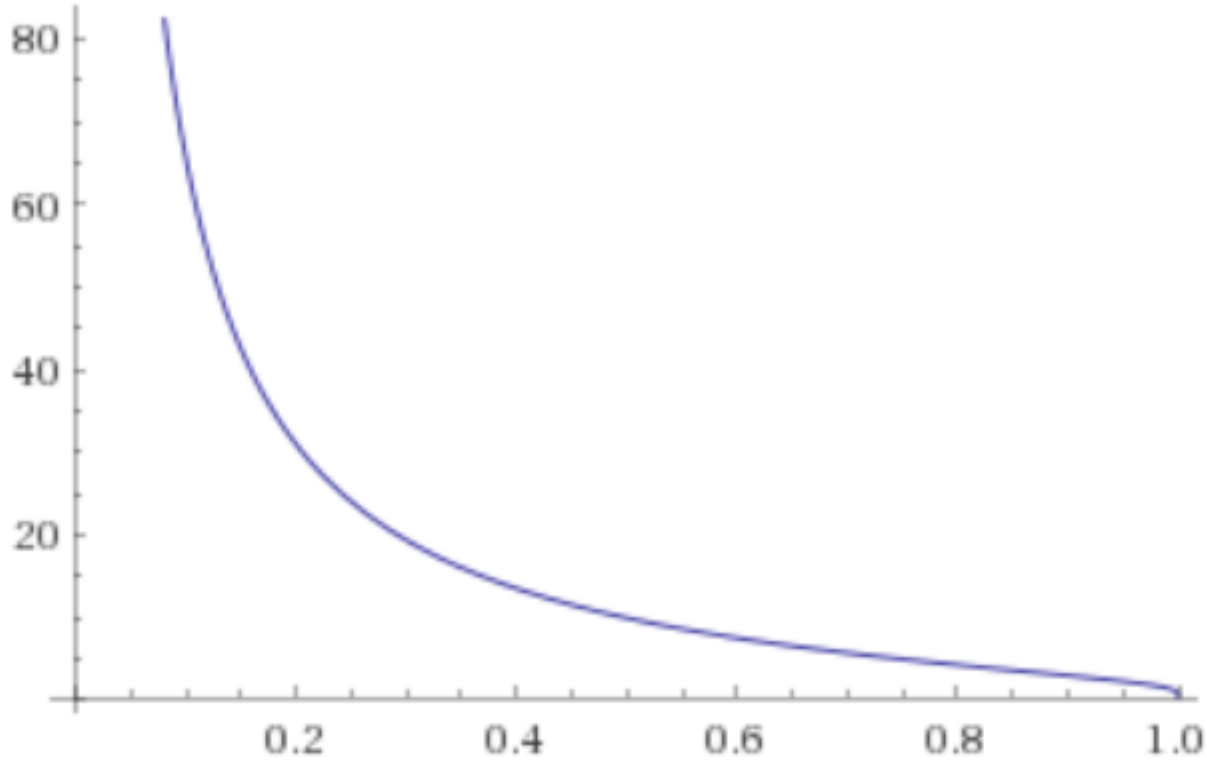
- **Cool, I guess… but what does this mean?**
- **Suppose P = 0.5 (we only have a 50% chance of success on any given run) and α = 0.001, we only tolerate a 0.1% error**
- **How many runs do we need to get this level of confidence?**
  - Only 10! This is a constant multiple

RANDOMIZATION

- In fact, suppose we always want our error to be 0.1%, how does this change with p?

RANDOMIZATION

- In fact, suppose we always want our error to be 0.1%, how does this change with p?



RANDOMIZATION

- Even if p is 0.1, only a 10% chance of success, we only need to run the algorithm 80 times to get a 0.001 confidence level

RANDOMIZATION

- Even if p is 0.1, only a 10% chance of success, we only need to run the algorithm 80 times to get a 0.001 confidence level
- What does this mean?

RANDOMIZATION

- **Even if p is 0.1, only a 10% chance of success, we only need to run the algorithm 80 times to get a 0.001 confidence level**
- **What does this mean?**
  - Randomized algorithms don't have to be complicated, if you can create a *reasonable* guess and can verify it in a short amount of time, then you can get good performance just from running repeatedly.

MINCUT

- Suppose there is a graph G(V,E)

MINCUT

- **Suppose there is a graph G(V,E)**
- **Find the two non-empty subgraphs V_1 and V_2 such that $V_1 \cup V_2 = V$ and the set of edges connecting them are minimal**

MINCUT

- Suppose there is a graph G(V,E)
- Find the two non-empty subgraphs V_1 and V_2 such that $V_1 \cup V_2 = V$ and the set of edges connecting them are minimal

- Why do we even care?

MINCUT

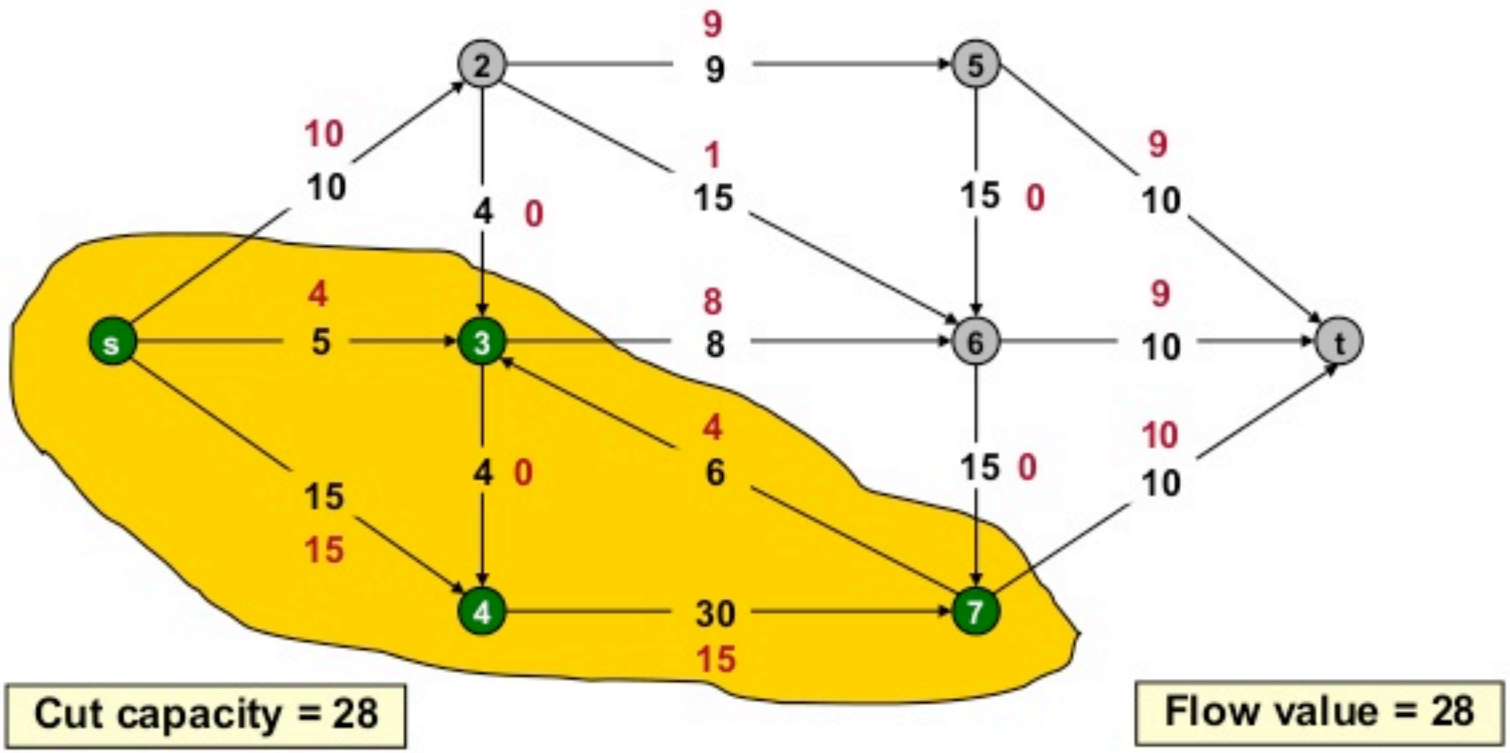
- **Suppose there is a graph G(V,E)**
- **Find the two non-empty subgraphs V_1 and V_2 such that $V_1 \cup V_2 = V$ and the set of edges connecting them are minimal**

- **Why do we even care?**
  - The min-cut is the maximum flow, if we are trying to connect two cities, the limit of traffic flow between nodes in the network

Max-Flow Min-Cut Theorem

MAX-FLOW MIN-CUT THEOREM (Ford-Fulkerson, 1956): In any network, the value of the max flow is equal to the value of the min cut.

- "Good characterization."
- Proof IOU.



FORD-FULKERSON

Algorithm [edit]

Let $G(V,E)$ be a graph, and for each edge from u to v, let $c(u,v)$ be the capacity and $f(u,v)$ be the flow. We want to find the maximum flow from the source s to the sink t. After every step in the algorithm the following is maintained:

Capacity constraints:	$\forall(u,v)\in E\ f(u,v)\le c(u,v)$	The flow along an edge can not exceed its capacity.
Skew symmetry:	$\forall(u,v)\in E\ f(u,v)=-f(v,u)$	The net flow from u to v must be the opposite of the net flow from v to u (see example).
Flow conservation:	$\forall u\in V: u\neq s \text{ and } u\neq t \Rightarrow \sum_{w\in V} f(u,w)=0$	That is, unless u is s or t. The net flow to a node is zero, except for the source, which "produces" flow, and the sink, which "consumes" flow.
Value(f):	$\sum_{(s,u)\in E} f(s,u)=\sum_{(v,t)\in E} f(v,t)$	That is, the flow leaving from s must be equal to the flow arriving at t.

This means that the flow through the network is a *legal flow* after each round in the algorithm. We define the **residual network** $G_f(V,E_f)$ to be the network with capacity $c_f(u,v)=c(u,v)-f(u,v)$ and no flow. Notice that it can happen that a flow from v to u is allowed in the residual network, though disallowed in the original network: if $f(u,v)>0$ and $c(v,u)=0$ then $c_f(v,u)=c(v,u)-f(v,u)=f(u,v)>0$.

Algorithm Ford–Fulkerson

Inputs Given a Network $G=(V,E)$ with flow capacity c, a source node s, and a sink node t

Output Compute a flow f from s to t of maximum value

1. $f(u,v)\leftarrow 0$ for all edges (u,v)
2. While there is a path p from s to t in G_f, such that $c_f(u,v)>0$ for all edges $(u,v)\in p$:
   1. Find $c_f(p)=\min\{c_f(u,v):(u,v)\in p\}$
   2. For each edge $(u,v)\in p$
      1. $f(u,v)\leftarrow f(u,v)+c_f(p)$ (*Send flow along the path*)
      2. $f(v,u)\leftarrow f(v,u)-c_f(p)$ (*The flow might be "returned" later*)

The path in step 2 can be found with for example a breadth-first search or a depth-first search in $G_f(V,E_f)$. If you use the former, the algorithm is called Edmonds–Karp.

FORD-FULKERSON

- Bleh. Garbage. Who has the time?

FORD-FULKERSON

- Bleh. Garbage. Who has the time?
- Can we estimate the min-cut?

FORD-FULKERSON

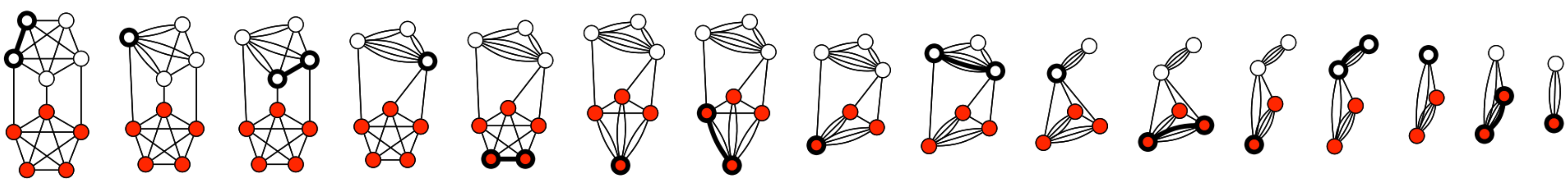
- **Bleh. Garbage. Who has the time?**
- **Can we estimate the min-cut?**
  - What might be an easy estimator?

FORD-FULKERSON

- **Bleh. Garbage. Who has the time?**
- **Can we estimate the min-cut?**
  - What might be an easy estimator?

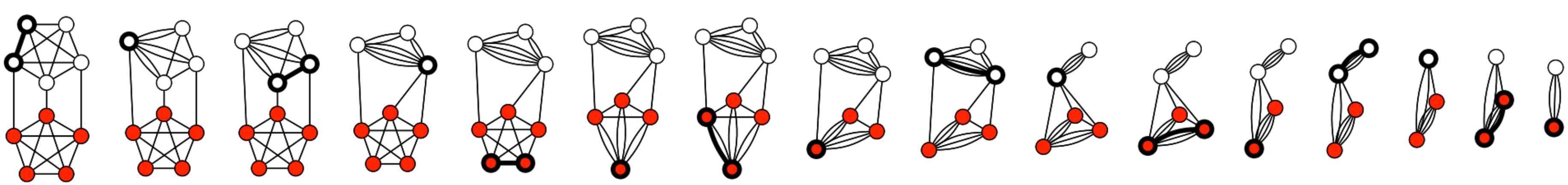
KARGER'S ALGORITHM

- **Bleh. Garbage. Who has the time?**
- **Can we estimate the min-cut?**
  - What might be an easy estimator?
- **Contract edges at random!**
  - How many edges will you contract to get two subgraphs?



KARGER'S ALGORITHM

- **Bleh. Garbage. Who has the time?**
- **Can we estimate the min-cut?**
  - What might be an easy estimator?
- **Contract edges at random!**
  - How many edges will you contract to get two subgraphs?
  - Only |V|-2



KARGER'S ALGORITHM

- Does this work?

KARGER'S ALGORITHM

- **Does this work?**
  - Success probability of 2/|E|

KARGER'S ALGORITHM

- **Does this work?**
  - Success probability of 2/|E|
  - Run it O(E) times, and you have a bounded success rate!

HARDWARE CONSTRAINTS

- So far, we've taken for granted that memory access in the computer is constant and easily accessible

HARDWARE CONSTRAINTS

- **So far, we've taken for granted that memory access in the computer is constant and easily accessible**
  - This isn't always true!

HARDWARE CONSTRAINTS

- **So far, we've taken for granted that memory access in the computer is constant and easily accessible**
  - This isn't always true!
  - At any given time, some memory might be cheaper and easier to access than others

HARDWARE CONSTRAINTS

- **So far, we’ve taken for granted that memory access in the computer is constant and easily accessible**
  - This isn’t always true!
  - At any given time, some memory might be cheaper and easier to access than others
  - Memory can’t always be accessed easily

HARDWARE CONSTRAINTS

- **So far, we've taken for granted that memory access in the computer is constant and easily accessible**
  - This isn't always true!
  - At any given time, some memory might be cheaper and easier to access than others
  - Memory can't always be accessed easily
  - Sometimes the OS lies, and says an object is "in memory" when it's actually on the disk

HARDWARE CONSTRAINTS

- **Back on 32-bit machines, each program had access to 4GB of memory**

HARDWARE CONSTRAINTS

- **Back on 32-bit machines, each program had access to 4GB of memory**
  - This isn’t feasible to provide!

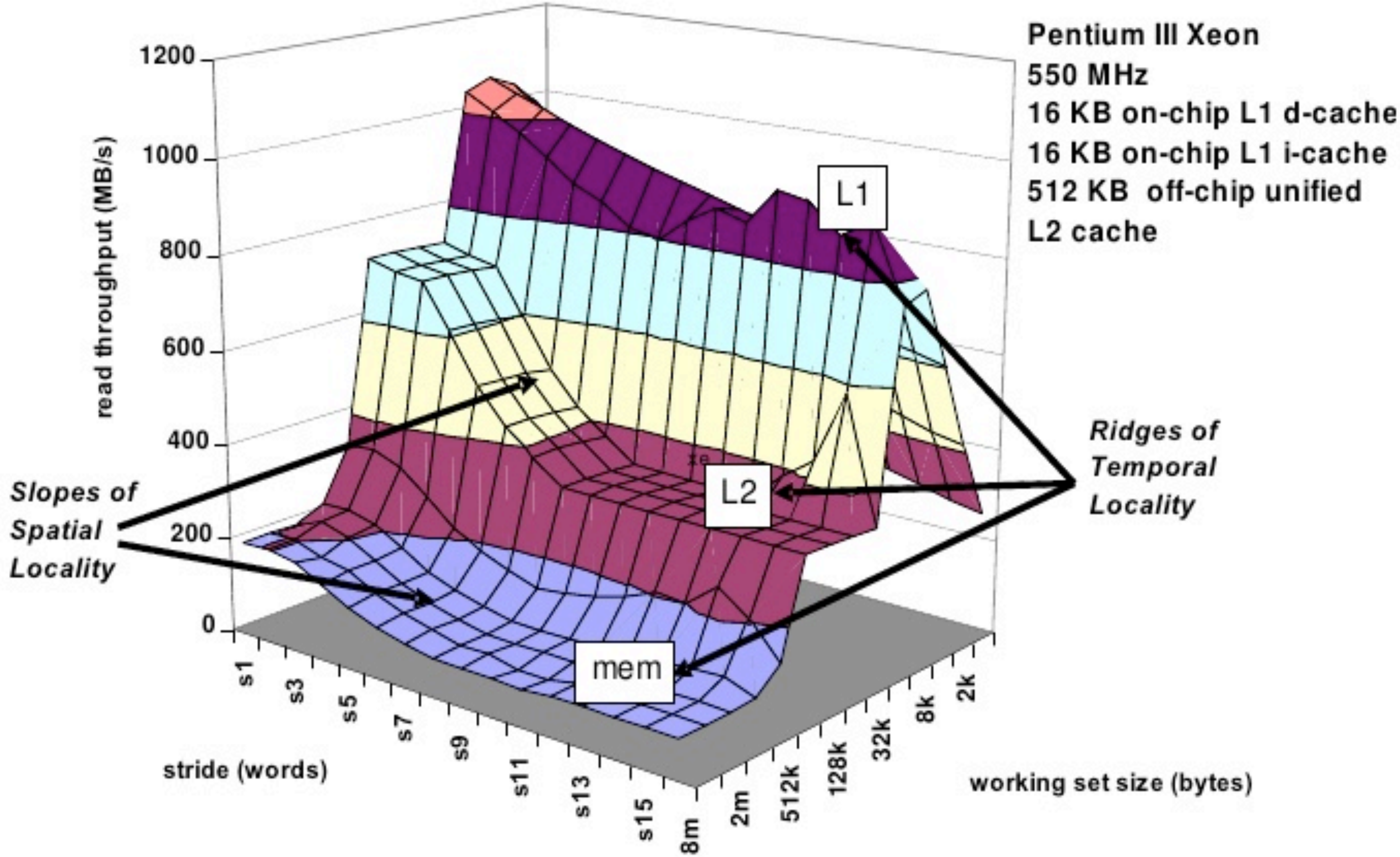
HARDWARE CONSTRAINTS

- **Back on 32-bit machines, each program had access to 4GB of memory**
  - This isn’t feasible to provide!
  - Sometimes there isn’t enough available, and so memory that hasn’t been used in a while gets pushed to the disk

HARDWARE CONSTRAINTS

- **Back on 32-bit machines, each program had access to 4GB of memory**
  - This isn't feasible to provide!
  - Sometimes there isn't enough available, and so memory that hasn't been used in a while gets pushed to the disk
- **Memory that is frequently accessed goes to the cache, which is even faster than RAM**

The Memory Mountain



LOCALITY AND PAGES

- **So, the OS does two smart things**
  - Spatial locality – if you use memory index Ox371347AB, you are likely to need Ox371347AC – bring both into cache
  - These are called pages, and they are usually around 4kb

LOCALITY AND PAGES

- **So, the OS does two smart things**
  - Spatial locality – if you use memory index Ox371347AB, you are likely to need Ox371347AC – bring both into cache
  - These are called pages, and they are usually around 4kb
  - All of the processes on your computer have access to pages in memory.

LOCALITY AND PAGES

- When you call new in Java, you are requesting new memory from the heap. If there isn’t memory there, the JVM needs to get new memory from the OS

LOCALITY AND PAGES

- **When you call new in Java, you are requesting new memory from the heap. If there isn't memory there, the JVM needs to get new memory from the OS**
  - The OS only uses memory in page sizes

LOCALITY AND PAGES

- **When you call new in Java, you are requesting new memory from the heap. If there isn't memory there, the JVM needs to get new memory from the OS**
  - The OS only uses memory in page sizes
  - So if you allocate 100Bytes of data, you overallocate to 4kb!

LOCALITY AND PAGES

- **When you call new in Java, you are requesting new memory from the heap. If there isn't memory there, the JVM needs to get new memory from the OS**
  - The OS only uses memory in page sizes
  - So if you allocate 100Bytes of data, you overallocate to 4kb!
  - But you can use that 4kb if you need more

LOCALITY AND PAGES

- **Secondly, the OS uses temporal locality,**

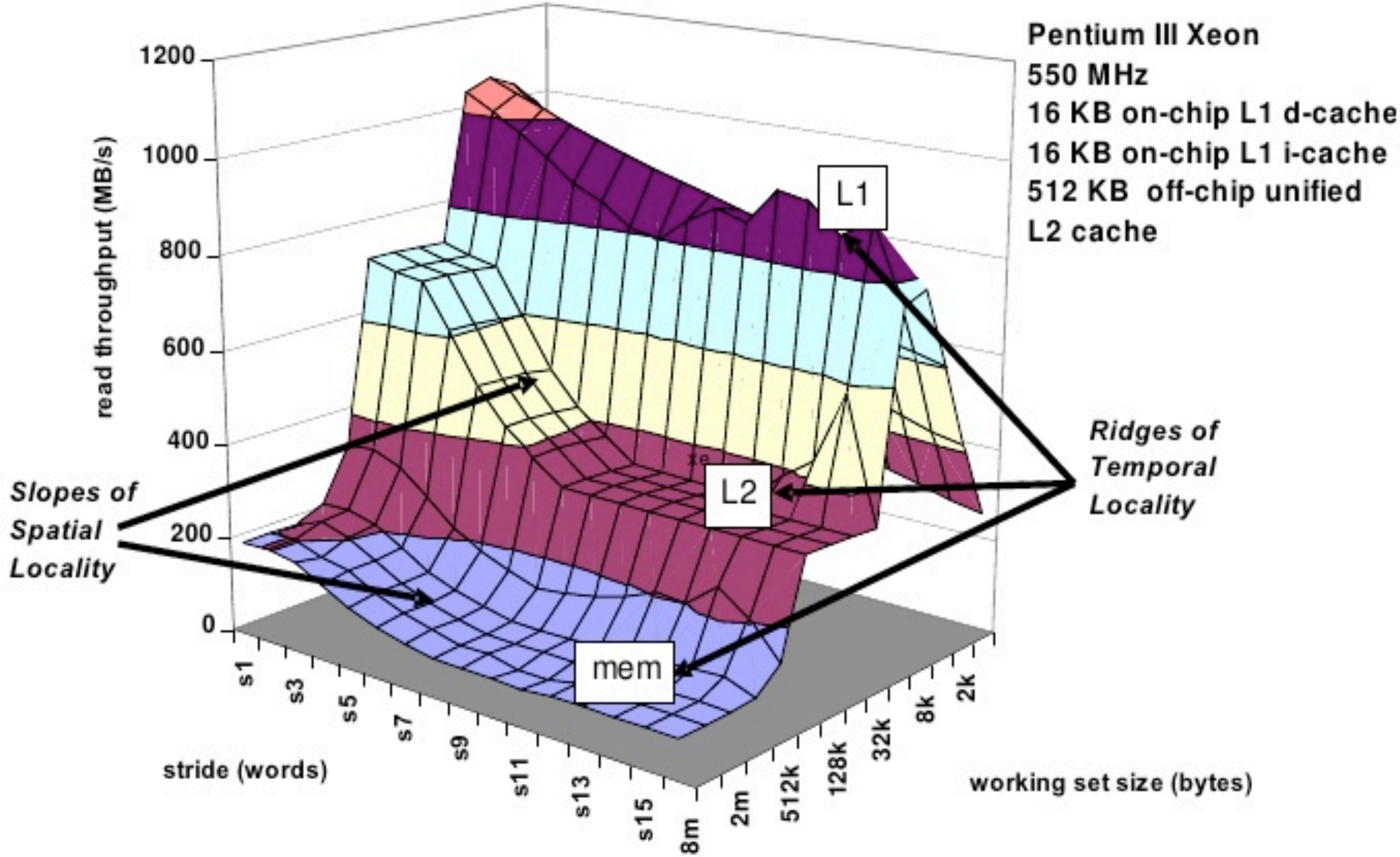
LOCALITY AND PAGES

- **Secondly, the OS uses temporal locality,**
  - Memory recently accessed is likely to be accessed again

LOCALITY AND PAGES

- **Secondly, the OS uses temporal locality,**
  - Memory recently accessed is likely to be accessed again
  - Bring recently used data into faster memory

The Memory Mountain



LOCALITY AND PAGES

- **Secondly, the OS uses temporal locality,**
  - Memory recently accessed is likely to be accessed again
  - Bring recently used data into faster memory
- **Types of memory (by speed)**
  - Register
  - L1,L2,L3
  - Memory
  - Disk
  - The interwebs (the cloud)

LOCALITY AND PAGES

- **The OS is always processing this information and deciding which is the best**
    - This is why arrays are faster in practice, they are always next to each other in memory

LOCALITY AND PAGES

- **The OS is always processing this information and deciding which is the best**
  - This is why arrays are faster in practice, they are always next to each other in memory
  - Each new node in a tree may not even be in the same page in memory!!

LOCALITY AND PAGES

- **The OS is always processing this information and deciding which is the best**
  - This is why arrays are faster in practice, they are always next to each other in memory
  - Each new node in a tree may not even be in the same page in memory!!
- **Important to consider when designing and explaining design problems.**

FRIDAY

- **Exam review during class**
- **Exam review EEB 105 4:30 – 6:00**