CSE 332

JULY 31ST – ALPHA BETA

ADMINISTRIVIA

- **P3 is out**
  - Partners form filled out by noon today!
  - Please nothing like last time

ADMINISTRIVIA

- **P3 is out**
    - Partners form filled out by noon today!
    - Please nothing like last time
- **3 Exercises out tonight, 2 due Friday one due Monday**

ADMINISTRIVIA

- **P3 is out**
  - Partners form filled out by noon today!
  - Please nothing like last time
- **3 Exercises out tonight, 2 due Friday one due Monday**

ADMINISTRIVIA

- **P3 is out**
  - Partners form filled out by noon today!
  - Please nothing like last time
- **3 Exercises out tonight, 2 due Friday one due Monday**
- **Considerations for final exam**
  - 1 hour is going to be difficult to cover all of the material

MINIMAX REVIEW

- Rules of the games

MINIMAX REVIEW

- **Rules of the games**
  - Two players

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum
  - Perfect information

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum
  - Perfect information
- **Works around a decision tree**

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum
  - Perfect information
- **Works around a decision tree**
  - Let’s look at a simple game… tic-toe

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum
  - Perfect information
- **Works around a decision tree**
  - Let’s look at a simple game… tic-toe
- **Players assume that the other team is playing optimally**

MINIMAX REVIEW

- **Rules of the games**
  - Two players
  - Zero sum
  - Perfect information
- **Works around a decision tree**
  - Let’s look at a simple game… tic-toe
- **Players assume that the other team is playing optimally**
  - Compute, what would I do if I was in the other persons shoes

MINIMAX REVIEW

- This strategy makes it easy to code games that fit into those parameters

MINIMAX REVIEW

- **This strategy makes it easy to code games that fit into those parameters**
    - Many games have very expansive decision trees

MINIMAX REVIEW

- **This strategy makes it easy to code games that fit into those parameters**
    - Many games have very expansive decision trees
    - How to improve?

MINIMAX REVIEW

- **This strategy makes it easy to code games that fit into those parameters**
  - Many games have very expansive decision trees
  - How to improve?
    - Parallelize or prune

MINIMAX REVIEW

- **This strategy makes it easy to code games that fit into those parameters**
  - Many games have very expansive decision trees
  - How to improve?
    - Parallelize or prune
- **How to parallelize minimax?**

MINIMAX REVIEW

- **This strategy makes it easy to code games that fit into those parameters**
  - Many games have very expansive decision trees
  - How to improve?
    - Parallelize or prune
- **How to parallelize minimax?**
  - Java uses the ForkJoinPool around RecursiveTasks, what are the important things the task needs to do and know?

MINIMAX REVIEW

- **Parallelizing Minimax**

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next
    - what moves are possible

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next
    - what moves are possible
  - Should return:

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next
    - what moves are possible
  - Should return:
    - The optimal move

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next
    - what moves are possible
  - Should return:
    - The optimal move
  - Other lessons?

MINIMAX REVIEW

- **Parallelizing Minimax**
  - RecursiveTask needs to know
    - the current state of the board
    - which player moves next
    - what moves are possible
  - Should return:
    - The optimal move
  - Other lessons?
    - The task should create other recursive tasks to find the results of the possible moves.

MINIMAX REVIEW

- How to parallelize a large number of processes?

MINIMAX REVIEW

- **How to parallelize a large number of processes?**
  - We know that we want the threads to start other threads
    - Divide and conquer

MINIMAX REVIEW

- **How to parallelize a large number of processes?**
  - We know that we want the threads to start other threads
    - Divide and conquer
  - There may also reach a point where we just want to allocate a bunch of threads serially

MINIMAX REVIEW

- **How to parallelize a large number of processes?**
  - We know that we want the threads to start other threads
    - Divide and conquer
  - There may also reach a point where we just want to allocate a bunch of threads serially
    - This is another cutoff

MINIMAX REVIEW

- **How to parallelize a large number of processes?**
  - We know that we want the threads to start other threads
    - Divide and conquer
  - There may also reach a point where we just want to allocate a bunch of threads serially
    - This is another cutoff
  - At the end, there is usually some base-case where the work is done sequentially

MINIMAX REVIEW

- **How to parallelize a large number of processes?**
  - We know that we want the threads to start other threads
    - Divide and conquer
  - There may also reach a point where we just want to allocate a bunch of threads serially
    - This is another cutoff
  - At the end, there is usually some base-case where the work is done sequentially
    - Could be multiple boards, or just taking the time to do multiple boards

MINIMAX REVIEW

- This is a new type of cutoff

MINIMAX REVIEW

- **This is a new type of cutoff**
  - Exercise due Friday involves you experimenting with the findPrimes parallel program we've given you, adding the forking cutoff and then running some experimentation
- **Parallelism aside, do we actually need to compute everything?**

MINIMAX REVIEW

- **This is a new type of cutoff**
    - Exercise due Friday involves you experimenting with the findPrimes parallel program we've given you, adding the forking cutoff and then running some experimentation
- **Parallelism aside, do we actually need to compute everything?**
    - No, we can perform alpha-beta pruning

MINIMAX REVIEW

- If we know our opponent has a better option, then they'll take it

MINIMAX REVIEW

- **If we know our opponent has a better option, then they'll take it**
    - [Alpha,beta] is a pair of bounds for acceptable end values for a particular move

MINIMAX REVIEW

- **If we know our opponent has a better option, then they'll take it**
  - [Alpha,beta] is a pair of bounds for acceptable end values for a particular move
  - At any given level of the tree, we can only set/prune on either alpha or beta

MINIMAX REVIEW

- **If we know our opponent has a better option, then they'll take it**
  - [Alpha,beta] is a pair of bounds for acceptable end values for a particular move
  - At any given level of the tree, we can only set/prune on either alpha or beta
  - Therefore, alpha and beta need to be switching throughout the tree

MINIMAX REVIEW

- **If we know our opponent has a better option, then they'll take it**
  - [Alpha,beta] is a pair of bounds for acceptable end values for a particular move
  - At any given level of the tree, we can only set/prune on either alpha or beta
  - Therefore, alpha and beta need to be switching throughout the tree
- **Cheating with Adam's slides**
  - https://courses.cs.washington.edu/courses/cse332/17wi/lectures/p3/p3.pdf

ALPHA BETA

- We can reduce the number of edges we need to consider in order to eliminate some nodes

ALPHA BETA

- **We can reduce the number of edges we need to consider in order to eliminate some nodes**
  - We will always have to consider all of the moves at some level of the tree

ALPHA BETA

- **We can reduce the number of edges we need to consider in order to eliminate some nodes**
    - We will always have to consider all of the moves at some level of the tree
    - The very first move needs to be recursively analyzed to the very bottom, then our alpha beta is [-inf,inf]

ALPHA BETA

- **We can reduce the number of edges we need to consider in order to eliminate some nodes**
  - We will always have to consider all of the moves at some level of the tree
  - The very first move needs to be recursively analyzed to the very bottom, then our alpha beta is [-inf,inf]
  - All of the moves at this first level need to be calculated, you can't know for sure that you can/cannot improve

ALPHA BETA

- **We can reduce the number of edges we need to consider in order to eliminate some nodes**
  - We will always have to consider all of the moves at some level of the tree
  - The very first move needs to be recursively analyzed to the very bottom, then our alpha beta is [-inf,inf]
  - All of the moves at this first level need to be calculated, you can’t know for sure that you can/cannot improve
  - Remember, then, the number of nodes alpha-beta can prune is dependent on the order that they are considered.

ALPHA BETA

- **We can reduce the number of edges we need to consider in order to eliminate some nodes**
  - We will always have to consider all of the moves at some level of the tree
  - The very first move needs to be recursively analyzed to the very bottom, then our alpha beta is [-inf,inf]
  - All of the moves at this first level need to be calculated, you can't know for sure that you can/cannot improve
  - Remember, then, the number of nodes alpha-beta can prune is dependent on the order that they are considered.
  - Move ordering is a good heuristic for p3 to save some time

ITERATIVE DEEPENING

- One final topic about P3

ITERATIVE DEEPENING

- **One final topic about P3**
  - Chess is a timed game, so you want to balance time spent with how much computing you'll need

ITERATIVE DEEPENING

- **One final topic about P3**
  - Chess is a timed game, so you want to balance time spent with how much computing you'll need
  - So, first try to run minimax/alphabeta at depth k, then if you have time, run minimax/alphabeta at depth k+1.

ITERATIVE DEEPENING

- **One final topic about P3**
  - Chess is a timed game, so you want to balance time spent with how much computing you'll need
  - So, first try to run minimax/alphabeta at depth k, then if you have time, run minimax/alphabeta at depth k+1.
  - We won't be having you compete against bots, but we will be having you compete against a timer, you can only have so much time per move.

PARALLEL PRIMATIVES

- So far we’ve seen two parallel primatives

PARALLEL PRIMATIVES

- **So far we've seen two parallel primatives**
  - Scan
    - Return some constant value from the whole array

PARALLEL PRIMATIVES

- **So far we've seen two parallel primatives**
  - Scan
    - Return some constant value from the whole array
  - Map

PARALLEL PRIMATIVES

- **So far we've seen two parallel primatives**
  - Scan
    - Return some constant value from the whole array
  - Map
    - Apply some function to each element in the array

PARALLEL PRIMATIVES

- **So far we've seen two parallel primatives**
  - Scan
    - Return some constant value from the whole array
  - Map
    - Apply some function to each element in the array
- **Together, these are powerful tools of parallelism, but they may not be sufficient**

PARALLEL PRIMATIVES

- We’re going to introduce two new types of problems

PARALLEL PRIMATIVES

- **We’re going to introduce two new types of problems**
  - Scan
    - Returns a modified array where each answer depends on the answer before it

PARALLEL PRIMATIVES

- **We're going to introduce two new types of problems**
  - Scan
    - Returns a modified array where each answer depends on the answer before it
  - Pack
    - Filter the array subject to some conditions

PARALLEL PRIMATIVES

- Scan

PARALLEL PRIMATIVES

- **Scan**
  - Given an array of integers, mutate the array so that each element contains the sum of the numbers up to that point

PARALLEL PRIMATIVES

- **Scan**
  - Given an array of integers, mutate the array so that each element contains the sum of the numbers up to that point
  - (i.e.) [1,2,3,4] becomes [1,3,6,10]

PARALLEL PRIMATIVES

- **Scan**
  - Given an array of integers, mutate the array so that each element contains the sum of the numbers up to that point
  - (i.e.) [1,2,3,4] becomes [1,3,6,10]
  - This is more complicated than a simple map, the function requires input from all the data before it.

PARALLEL PRIMATIVES

- **Scan**
  - Given an array of integers, mutate the array so that each element contains the sum of the numbers up to that point
  - (i.e.) [1,2,3,4] becomes [1,3,6,10]
  - This is more complicated than a simple map, the function requires input from all the data before it.
  - What are some ways we can parallelize this process?

PARALLEL PRIMATIVES

- **Scan**
  - Given an array of integers, mutate the array so that each element contains the sum of the numbers up to that point
  - (i.e.) [1,2,3,4] becomes [1,3,6,10]
  - This is more complicated than a simple map, the function requires input from all the data before it.
  - What are some ways we can parallelize this process?
    - How do you find the value of a particular node?

PARALLEL PRIMATIVES

- Partial sum problem

PARALLEL PRIMATIVES

- **Partial sum problem**
    - Each node needs information from all the numbers before it

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?
  - What is the actual function?

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?
  - What is the actual function?
    - Value is going to be the presum + the current value

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?
  - What is the actual function?
    - Value is going to be the presum + the current value
    - These presum values are going to be reused!

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?
  - What is the actual function?
    - Value is going to be the presum + the current value
    - These presum values are going to be reused!
    - Think about applying a sum reduce!

PARALLEL PRIMATIVES

- **Partial sum problem**
  - Each node needs information from all the numbers before it
  - How to parallelize?
    - What are some ideas?
  - What is the actual function?
    - Value is going to be the presum + the current value
    - These presum values are going to be reused!
    - How would you apply a sum reduce!
  - Scan trees!