CS F364
Design & Analysis of Algorithms

RANDOMIZED ALGORITHMS

Game Tree Evaluation

A Deterministic Algorithm

A Randomized Algorithm

Game Trees and Evaluation

- A game tree is a rooted tree in which
  - internal nodes are labeled MIN or MAX at alternate levels and
  - a value – a number - is associated with each leaf
- Evaluation of a game tree:
  - Each leaf returns its value
  - Each internal node
    - *if labeled MAX* returns the largest value returned by its children and
    - *if labeled MIN* returns the smallest value returned by its children
- Game Tree evaluation is useful in AI - particularly in game playing programs
  - Examples: Chess, Tic-Tac-Toe

Game Tree Evaluation

- Consider the following special cases:
  - Boolean game trees
    - Values at the leaves are bits i.e. o or 1
    - MIN=AND, MAX=OR
  - $T_{d,k}$ trees:
    - Every internal node has exactly d children and
    - Every leaf node is at distance exactly 2*k from the root

Evaluation of Boolean $T_{d,k}$ trees

- An instance of the evaluation problem consists of
  - the tree $T_{d,k}$ for a fixed d and a fixed k, and
  - a Boolean value for each leaf.
- Number of leaves in $T_{d,k}$ tree =?
- How does evaluation proceed?

Evaluation of Boolean $T_{d,k}$ trees

- Deterministic algorithms:
  - Choice of next step i.e. next child to be evaluated is a deterministic function of the evaluations so far
- Claim:
  - For any deterministic algorithm, there is an instance of the $T_{d,k}$ tree that forces the algorithm to read all d^{2*k} leaves.
- In the special case of $d=2$, the cost of the deterministic algorithm would be 4^k
  - Note that we are counting "the number of leaves read" as the cost as this is the dominant factor in the running time of an algorithm

Evaluation of Boolean $T_{2,k}$ trees

- A randomized approach:
  - Consider a single AND node with two leaves.
  - If this node were to return 0 at least one child must return 0
  - A deterministic algorithm inspects the leaves in a fixed order
    - i.e. an adversary may hide the 0 at the second leaf leading to a worst case scenario
  - Reading the leaves in a random order foils such an adversary
    - With probability ½ a randomized algorithm would choose the second leaf
      - i.e. expected number of steps $= 1/2 * 1 + 1/2 * 2 = 3/2$ (instead of 2 for a deterministic algorithm)

Evaluation of Boolean $T_{2,k}$ trees

- A randomized approach (contd.):
  - A similar approach will work for an OR node that must return 1
  - But what about an AND node returning 1 or an OR node returning 0 ?
    - For such nodes, this approach does not improve the odds if the leaves are children.
  - But consider an internal AND node that must return 1:
    - Each child is an OR node and must each return 1
      - i.e. a randomized approach will improve the performance in evaluating them

Evaluation of Boolean $T_{2,k}$ trees

- A randomized algorithm:
  - To evaluate an AND node v
    - Choose one of v's children randomly and evaluate recursively
    - If that child returns
      - 1 : evaluate the other child
      - 0 : return 0 for v
  - To evaluate an OR node v
    - Choose one of v's children randomly and evaluate recursively
    - If that child returns
      - 0 : evaluate the other child
      - 1 : return 1 for v

Evaluation of Boolean $T_{2,k}$ trees

- Cost of evaluation
  - Claim: Expected cost of evaluating any instance of $T_{2,k}$ using the randomized algorithm is at most 3^k
  - Proof: By induction on k
    - Basis: (k=1) Trivial
    - Induction Step:
      - Assume that the expected cost of evaluating any instance of $T_{2,k-1}$ is at most 3^{k-1}
      - Assume that T is rooted at an AND node (the other case is symmetric)
        - In this case T has two children, say T1 and T2 each rooted at an OR node.

Evaluation of Boolean $T_{2,k}$ trees

- Cost of evaluation (contd.):
  - Induction Step (contd.): T1 is rooted at an OR node with 2 children – each of which is a $T_{2,k-1}$ tree
    - If T1 were to return 1 at least one of its children must return 1
      - With probability ½ this child is chosen first incurring a cost of at most 3^{k-1} (by Hypothesis)
      - With probability at most ½ both sub trees are evaluated incurring a cost of $2 * 3^{k-1}$ (by Hypothesis)
      - i.e. the expected cost is $½ * 3^{k-1} + ½ * 2 * 3^{k-1} = 3/2 * 3^{k-1}$
    - If T1 were to return 0 both children must be evaluated, incurring a cost of at most $2 * 3^{k-1}$

Evaluation of Boolean $T_{2,k}$ trees

- Cost of evaluation (contd.):
  - Induction Step (contd.): T is rooted at an AND node with 2 children - say T1 and T2 – both are rooted at OR node
  - Case 1: *If T evaluates to 1* then both its sub-trees rooted at OR nodes return 1
    - Using the previous result (for OR-nodes evaluating to 1) and linearity of expectation,
      - the expected cost of evaluating $T_{2,k}$ to 1 is at most $2 * 3/2 * 3^{k-1} = 3^k$

Evaluation of Boolean $T_{2,k}$ trees

- Cost of evaluation (contd.):
  - Induction Step (contd.): T is rooted at an AND node with 2 children - say T1 and T2 – both are rooted at OR node
  - Case 0: *If T evaluates to 0,* at least one of its sub-trees must return 0
    - With probability at least ½ it is chosen first, in which case the cost is
      - $2 * 3^{k-1}$ using the previous result (for OR-nodes evaluating to 0)
    - With probability ½ it is chosen second i.e. two OR-nodes are to be evaluated, in which case the cost is
      - $2 * 2 * 3^{k-1}$ using the previous result (for OR-nodes evaluating to 0)
    - So the expected cost is at most $1/2 * 2 * 3^{k-1} + 1/2 * 2 * 2 * 3^{k-1} = 3^k$

Evaluation of Boolean $T_{2,k}$ trees

- Cost of evaluation
  - Claim: Expected cost of evaluating any instance of $T_{2,k}$ using the randomized algorithm is at most 3^k
    - for a tree of size $n = 4^k$
      - i.e. expected cost is at most n ^ $\log_4 3$ i.e. $n^{0.793}$