Randomized Algorithms

Sariel Har-Peled

August 29, 2002

Intro

Quicksort

Items $S_1, \ldots, S_n$ to be sorted

- suppose could pick middle element:

$$T(n) = 2T(n/2) + O(n) = O(n \log n)$$

works since divides into much smaller subproblems

- picking middle is hard. But an almost middle element is OK.

- pick random element. “probably” near middle and divides problem in two

- bound expected number of comparisons $C$

$X_{ij} = 1$ if compare $i$ to $j$

- linearity of expectation: $E[C] = \sum E[X_{ij}]$

- $E[X_{ij}] = p_{ij}$

- Consider smallest recursive call involving both $i$ and $j$.

- pivot must be one of $S_i, \ldots, S_j$. all equally likely

- $S_i$ and $S_j$ get compared if pivot is $S_i$ or $S_j$

- probability is at most $2/(j - i + 1)$ (may have outer elements)
• analysis:

\[
\sum_{i=1}^{n} \sum_{j>i}^{n} p_{ij} \leq \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{2}{j - i + 1} \\
= \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \\
\leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \\
\leq 2n H_n
\]

(Define \( H_n \), claim \( O(\log n) \).

\[ = O(n \log n). \]

• analysis holds for every input, doesn’t assume random input
• we proved expected. can show high probability
• how did we pick a random elements?
• algorithm always works, but might be slow.

**BSP**

• linearity of expectation.
• Rendering an image
  - render a collection of polygons (lines)
  - painters algorithm: draw from back to front; let front overwrite
  - need to figure out order with respect to user
• define BSP.
  - BSP is a data structure that makes order determination easy
  - Build in preprocess step, then render fast.
Choose any hyperplane (root of tree), split lines onto correct side of hyperplane, recurse
- If user is on side 1 of hyperplane, then nothing on side 2 blocks side 1, so paint it first. Recurse.
- time=BSP size

• sometimes must split to build BSP
• how limit splits?
• autopartitions
• random auto
• analysis
  - \( \text{index}(u, v) = k \) if \( k \) lines block \( v \) from \( u \)
  - \( u \vdash v \) if \( v \) cut by \( u \) auto
  - probability \( 1/(1 + \text{index}(u, v)) \).
  - tree size is (by linearity of \( E \))
    \[ n + \sum 1/\text{index}(u, v) \leq \sum_u 2H_n \]

• result: \textbf{exists} size \( O(n \log n) \) auto
• gives randomized construction
• equally important, gives \textbf{probabilistic existence proof} of a small BSP
• so might hope to find deterministically.

3