Quicksort

• Quicksort follows the „Divide and Conquer“ paradigm
• The algorithm is best described recursively

• Idea:
  – Split the sequence into two
    • All elements in one sequence are smaller than in the other
  – Sort each sequence
  – Put them back together
Quicksort

- Quicksort(A,l,r)
  - If l=r return A
  - Choose a pivot position j between l and r
  - u=1,v=1, initialize arrays B,C
  - Run Quicksort(B,1,u) and Quicksort(C,1,v) and return their output (concatenated), with A[j] in the middle
How fast is it?

• The quality of the algorithm depends on how we split up the sequence

• Intuition:
  – Even split will be best

• Questions:
  – What is the asymptotic running time?
  – Are approximately even splits good enough?
Worst Case Time

• We look at the case when we really just split into the pivot and the rest (maximally uneven)
• Let $T(n)$ denote the number of comparisons for $n$ elements
• $T(2)=1$
• $T(n)<T(n-1)+n-1$
• Solving the recurrence gives $T(n)=O(n^2)$
Best Case Time

• Every pivot splits the sequence in half

• $T(2)=1$
• $T(n)=2T(n/2)+n-1$

• Questions:
  – How to solve this?
  – What if the split is $3/4$ vs. $1/4$ ?
Recurrences

• How to solve simple recurrences
• Several techniques
• Idea: Consider the recursion tree
• For Quicksort every call of the procedure generates two calls to a smaller Quicksort procedure
  – Problem size 1 is solved immediately
• Nodes of the tree are labelled with the sequences that are sorted at that node
• The cost of a node is the number of comparisons used to split the sequence at the node. I.e. is equal to the length of the sequence at the node.
\[ [A_{[1]}, \ldots, A_{[\frac{n}{2}]}] \quad \bullet \quad [A_{[\nu_2+1]}, \ldots, A_{[n]}] \]

\[ \quad \bullet \quad [A_{[\nu]}] \]

\[ \quad \bullet \quad [A_{[\nu]}] \]

\[ \quad \bullet \quad [A_{[\nu]}] \]
Example: the perfect tree

• In the best case the sequence length halves-> after log n calls the sequence has length 1

• Depth of the tree is log n
  – Number of nodes is O(n)
  – But each node has a cost
    • nodes on level 1 cost n, one level 2 cost n/2 etc.
    – Level i has $2^i$ nodes of cost $n/2^i$

• Total cost is O(n) per level-> O(n log n)
Verifying the guess

• Guess: \( T(n) \leq n \log n \)
• \( T(2): T(n) = 1 \)
• \( T(n) = 2T(n/2) + n - 1 \)
  \[ \leq n \log(n/2) + n - 1 \]
  \[ = n \log n - n + n - 1 \]
  \[ \leq n \log n \]
The Master Theorem

• The Master Theorem is a way to get solutions to recurrences

• **Theorem:**
  a,b constant, f(n) function
  Recurrence T(n)=aT(n/b)+f(n)

- 1) \( f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a}) \)
- 2) \( f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a \cdot \log \log n}) \)
- 3) \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \Rightarrow T(n) = \Theta(f(n)) \)
The Master Theorem

• We omit the proof
• Application:
  • $T(n) = 9T(n/3) + n$
  • $a = 9$, $b = 3$, $f(n) = n$, $n^{\log_b(a)} = n^2$
    - Case 1 applies, Solution is $T(n) = O(n^2)$
Attempt on the case of uneven splits

• Assume every pivot splits exactly $\frac{3}{4}n$ vs. $n/4$
  – $T(n)=T(3n/4)+T(n/4)+n$

• Same idea:
  – Nodes on level $i$ have cost $n/(3/4)^i$ at most
  – There are at most $\log_{4/3} n$ levels
  – What is the total cost of all nodes at a level?
  – Note that all nodes on 1 level correspond to a partition of all $n$ inputs!
  – less than $n$ comparisons on one level
Quicksort Time

• So if every split is partitioning the sequence somewhat evenly (99% against 1%) then the running time is $O(n \log n)$
Average Case Time

• Suppose the pivot is chosen in any way
  – Say, the first element
• Claim: the expected running time of Quicksort is $O(n \log n)$
  – Expected over what?
  – Chosing a *random* permutation as the input
    • Recall that the input to the sorting problem is a permutation
Average Time

• Intuition: Most of the time the first element will be in the “middle” of the sequence for a random permutation
  – Most of the time we have a (quite) balanced split

• Constant probability of an uneven split
  – Can increase running time by a constant factor only
    • Assume nothing gets done on those splits
    • “Merge” balanced and unbalanced splits
Average Time

• **Theorem:**
  On a uniformly random permutation the expected running time of Quicksort is $O(n \log n)$
Note of Caution

• For any fixed (simple) pivoting rule there are still permutations that need time \( n^2 \)
  – e.g. pivot is always the minimum
• How to fix this?
• Choose pivot such that the algorithm behaves in the same way as for a random permutation!
Randomized Algorithms

• A randomized algorithm is an algorithm that has access to a source of random numbers
• Different types:
  – Measure expected running time
    • with respect to the random numbers, NOT the inputs
  – Allow errors with low probability
• We will (for now) consider the first type
Randomized Quicksort

• Use the standard Quicksort,
• BUT choose a random position between \( l \) and \( r \) as the pivot

• **Theorem:** Randomized Quicksort has (expected) running time \( O(n \log n) \)
Average Time

• The theorem about randomized Quicksort implies the theorem about the average case time bound for deterministic Quicksort

• Reason:
  – In any partition step the first element of a random permutation and an element for a fixed permutation behave in the same way
Proof

• Proof (randomized Quicksort)
• We will count the expected number of comparisons
• Denote by $X_{ij}$ the indicator random variable that is 1 if $x_i$ is compared to $x_j$
  – At any time
  – $x_i$ is the $i$th element of the sorted sequence
• Note that all comparisons involve the pivot element
Proof

• The expected number of comparisons is
  \[ E[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_{ij}] \]

• \( E[X_{ij}] \) is the probability that \( x_i \) is compared to \( x_j \)

• \( Z_{ij} \) is the set of keys between \( x_i \) and \( x_j \)

• Claim:
  \( x_i \) is compared with \( x_j \) iff \( x_i \) or \( x_j \) is the first pivot chosen among the elements of \( Z_{ij} \)
Proof

- Pivots are random, i.e.,
  \( \text{Prob}(x_i \text{ first pivot in } Z_{ij}) = \frac{1}{j-i+1} \)
- \( E[X_{ij}] \leq \frac{2}{j-i+1} \)
- Number of comparisons:
  \[
  \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]
  \leq 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1}
  \leq 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k+1}
  < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}
  = O(n \log n) \quad \text{[Harmonic Series]} \]
Proof

• Hence the expected number of comparisons is $O(n \log n)$
• Easy to see that also the running time is $O(n \log n)$