Outline

- A lower bound for comparison-based sorting
- Linear time sorting

Run Time Bounds

Upper bounds on the \textit{Worst Case} running times of sorting algorithms:
- Bubble-sort: $O(n^2)$
- Modified Bubble-sort: $O(n^2)$
- Insertion-sort: $O(n^2)$
- Merge-sort: $O(n \log(n))$
- Heap-sort: $O(n \log(n))$
- Quick-sort: $O(n^2)$

Run Time Bounds cont.

Lower bounds on the \textit{Worst Case} running times of sorting algorithms:
- Bubble-sort: $\Omega(n^2)$
- Modified Bubble-sort: $\Omega(n^2)$
- Insertion-sort: $\Omega(n^2)$
- Merge-sort: $\Omega(n \log(n))$
- Heap-sort: $\Omega(n \log(n))$
- Quick-sort: $\Omega(n^2)$

Comparison-based sorting

A comparison-based sorting algorithm is one that doesn’t need to read the input, provided it is given the size of the input and a comparison oracle.

Lower Bound for Sorting

Theorem: Any comparison-based sorting algorithms has a worst-case running time that is $\Omega(n \log(n))$.

Proof of Theorem

A decision tree describes the queries of a comparison-based algorithm on input size $n$. A root to leaf path represents the sequence of queries for a particular input.
Proof of Theorem cont.

Each leaf corresponds to the permutation that sorts the input.

Proof cont.

- There must be at least \( n! \) leaves.
- A binary tree with \( n! \) leaves has a path with length at least \( \log(n!) \).
- By Stirling’s approximation
  \[ \log(n!) = \Omega(n \log(n)) \]

Outline

- A lower bound for comparison-based sorting
- Linear time sorting

Sorting in \( O(n) \)

Can we do it?

Yes – but only if we can make some assumptions about the input.

Some examples:
- Counting-sort
- Radix-sort
- Bucket-sort

Counting-sort(S)

- Assumption: The input integers are in the range \([0..B]\), where \( B \) is some constant.
- For \( i = 0 \) to \( B \): Count\((i) = 0\)
- For \( i = 1 \) to \( n \): Count\((S(i))++\)
- For \( i = 1 \) to \( B \): For \( j = 1 \) to count\((i)\): Write \( i \)

Counting-sort

- Is it correct?
  - Yes

- Is it fast?
  - \( O(n+B) \) but \( B \) is constant so \( O(n) \)
Radix-sort

- Assumption: The input are decimal integers with exactly D digits, where D is some constant.

- For $i = 1$ to $D$
  - Use counting-sort to sort $S$ based on $i^{th}$ digit using current order to break ties. (Least significant digit is digit 1.)

Radix-sort($S$)

| 329 | 720 | 720 | 329  |
| 457 | 355 | 329 | 355  |
| 657 | 436 | 436 | 436  |
| 839 | 457 | 839 | 457  |
| 436 | 657 | 355 | 657  |
| 720 | 329 | 457 | 720  |
| 355 | 839 | 657 | 839  |

Radix-sort

- Is it fast?
  - $O(Dn)$ but $D$ is constant so $O(n)$

- Is it correct?
  - Loop invariant: The red numbers are sorted.

Bucket-sort($S$)

- Assumption: The input are uniformly distributed in the range $[0..1)$.

- Initialize buckets $B[0..n-1]$ to nil

- For $i = 1$ to $n$
  - Insert $S[i]$ into the sorted list at $B \lfloor n \cdot S[i] \rfloor$
  - Output the concatenated buckets

Bucket-sort

- Is it correct?
  - Yes

- Is it fast?
  - The expected running time is $O(n)$. 
Average-case analysis

What does average-case mean?
- Deterministic algorithm with a known input distribution
- Randomized algorithm on any (i.e. worst-case) input

A brief tour of (discrete) probability theory…

- Sample space and elementary events
- Discrete probability distributions
- Discrete random variables
- Expectation

The experiment

- A fair coin is flipped
- Sample space: \{Head, Tail\}

The experiment

- Two fair coins are flipped
- Sample space: \{HH, HT, TH, TT\}

Discrete Probability Distribution

Assigns a real number to outcomes:
- Experiment 1: \(P(H) = P(T) = 1/2\)
- Experiment 2:
  \(P(HH) = P(HT) = P(TH) = P(TT) = 1/4\)

Discrete Probability Distribution

- \(P(A) \geq 0\)
- \(P(S) = 1\)
- \(P(A \cup B) = P(A) + P(B)\) when A and B are disjoint events.
Discrete Random Variable X

- Sample space is a finite set of real numbers.
- $X$ is the outcome of the experiment
- Probability distribution: $P(X=x)$
- Expected value of $X$:
  $$E[X] = \sum_{x \in S} x P(X=x)$$
  $$E[X^2] = \sum_{x \in S} x^2 P(X=x)$$
  $$\text{Var}(X) = E[(X-E[X])^2]$$

Example

- A unfair coin is tossed $n$ times:
  $P(H)=p$, $P(T)=1-p=q$
- $X$ is the number of heads
- $P(X=k) = \binom{n}{k} p^k q^{n-k}$ (Binomial distribution)
- $E[X] = np$
- $E[X^2] = npq + n^2 p^2$

Bucket-sort($S$)

- Let $T$ be the number of comparisons made by the algorithm on a random input of size $n$.
- Claim: $E[T] = O(n)$

Analysis

- Let $X_i$ be the number of input that belong in bucket $i$. $X_i$ is binomially distributed with $p=1/n$.
- $T \leq c \sum_{i=0}^{n-1} X_i^2$
- $E[T] \leq c \sum_{i=0}^{n-1} E[X_i^2]$
  $$= c \sum_{i=0}^{n-1} (npq + n^2 p^2) = O(n).$$