The Design and Analysis of Algorithms

Lecture 8 Divide and Conquer I

Zhenbo Wang

Department of Mathematical Sciences, Tsinghua University

Content

Mergesort

Counting Inversions

Closest Pair of Points

Divide and Conquer

- Divide-and-conquer.

  Divide up problem into several subproblems.

  Solve each subproblem recursively.

  Combine solutions to subproblems into overall solution.

- Most common usage.

  Divide problem of size n into two subproblems of size $n/2$ in linear time.

  Solve two subproblems recursively.

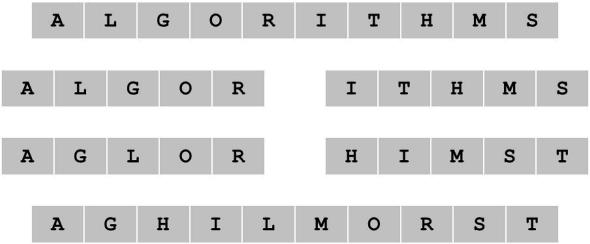
  Combine two solutions into overall solution in linear time.

Sorting Problem

- *Sorting problem*: Given a list of n elements from a totally-ordered universe, rearrange them in ascending order.
- Obvious applications: List names in a phone book; display Google PageRank results; list scores in a course $\cdots$
- Some problems become easier once elements are sorted: Binary search in a database; find the median $\cdots$
- Non-obvious applications: Convex hull; closest pair of points; interval scheduling; minimum spanning trees $\cdots$

Mergesort

- Recursively sort left half.
- Recursively sort right half.
- Merge two halves to make sorted whole.



Merging

Goal. Combine two sorted lists A and B into a sorted whole C.

- Scan A and B from left to right.
- Compare a_i and b_j.
- If $a_i \leq b_j$, append a_i to C (no larger than any remaining element in B).
- If $a_i > b_j$, append b_j to C (smaller than every remaining element in A).
- Demo

A Useful Recurrence Relation

Def. $T(n) =$ max number of compares to mergesort a list of size $\leq n$.

Note. $T(n)$ is monotone nondecreasing.

Mergesort recurrence.

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n & \text{otherwise} \end{cases}$$

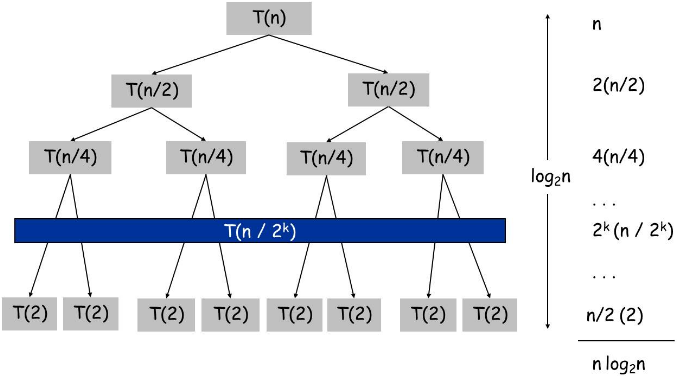
Solution. $T(n)$ is $O(n \log n)$.

We describe several ways to prove this recurrence.

- Initially we assume n is a power of 2 and replace $\leq$ with $=$.

Proof by Recursion Tree

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2T(n/2) + n & \text{otherwise} \end{cases}$$



Proof by Induction

Proposition. If $T(n)$ satisfies the following recurrence, then $T(n) = n \log n$.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2T(n/2) + n & \text{otherwise} \end{cases}$$

Pf 2. [by induction on n]

Base case: when $n = 1$, $T(1) = 0$.

Inductive hypothesis: assume $T(n) = n \log n$.

Goal: Show that $T(2n) = 2n \log(2n)$.

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \\ &= 2n \log(2n). \ \square \end{aligned}$$

Analysis of Mergesort Recurrence

Claim. If $T(n)$ satisfies the following recurrence, then $T(n) \leq n\lceil \log n \rceil$.

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n & \text{otherwise} \end{cases}$$

Pf. [by induction on n]

Base case: $n = 1$.

Define $n_1 = \lfloor n/2 \rfloor$ and $n_2 = \lceil n/2 \rceil$.

Induction step: assume true for $1, 2, \cdots, n-1$.

$$\begin{aligned} T(n) &\leq T(n_1) + T(n_2) + n \\ &\leq n_1\lceil \log n_1 \rceil + n_2\lceil \log n_2 \rceil + n \\ &\leq n\lceil \log n_2 \rceil + n \\ &\leq n(\lceil \log n \rceil - 1) + n = n\lceil \log n \rceil. \ \square \end{aligned}$$

Counting Inversions

- Music site tries to match your song preferences with others.

  You rank n songs.

  Music site consults database to find people with similar tastes.

- *Similarity metric*: Number of inversions between two rankings.

  My rank: $1, 2, \cdots, n$.

  Your rank: $a_1, a_2, \cdots, a_n$.

  Songs i and j are inverted if $i < j$, but $a_i > a_j$.

Songs

	A	B	C	D	E
Me	1	2	3	4	5
You	1	3	4	2	5

<u>Inversions</u>

3-2, 4-2

Applications

- Voting theory.
- Measuring the “sortedness” of an array.
- Sensitivity analysis of Google’s ranking function.
- Rank aggregation for meta-searching on the Web.
- Nonparametric statistics.

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two halves A and B.
- Conquer: recursively count inversions in each list.
- Combine: count inversions (a, b) with $a \in A$ and $b \in B$, and return sum of three counts.

1	5	4	8	10	2	6	9	12	11	3	7

Divide: $O(1)$.

1	5	4	8	10	2

6	9	12	11	3	7

1	5	4	8	10	2	6	9	12	11	3	7

Divide: $O(1)$.

1	5	4	8	10	2

6	9	12	11	3	7

Conquer: $2T(n / 2)$

5 blue-blue inversions

5-4, 5-2, 4-2, 8-2, 10-2

8 green-green inversions

6-3, 9-3, 9-7, 12-3, 12-7, 12-11, 11-3, 11-7

9 blue-green inversions

5-3, 4-3, 8-6, 8-3, 8-7, 10-6, 10-9, 10-3, 10-7

Combine: ???

Total = 5 + 8 + 9 = 22.

Counting Inversions: Combine

- Combine: Count blue-green inversions.

  Assume each half is sorted.

  Count inversions where a_i and a_j are in different halves.

  Merge two sorted halves into sorted whole.

  $T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \Rightarrow T(n) = O(n \log n)$.

3	7	10	14	18	19

2	11	16	17	23	25
6	3	2	2	0	0

13 blue-green inversions: 6 + 3 + 2 + 2 + 0 + 0

Count: O(n)

2	3	7	10	11	14	16	17	18	19	23	25

Merge: O(n)

Counting Inversions: Implementation

Input. List L.

Output. Number of inversions in L and sorted list of elements L'.

SORT – AND – COUNT(L)

```
if list L has one element then
    return (0, L).
end if
DIVIDE the list into two halves A and B.
(r_A, A) ← SORT-AND-COUNT(A).
(r_B, B) ← SORT-AND-COUNT(B).
(r_AB, L') ← MERGE-AND-COUNT(A, B).
return r_A + r_B + r_AB, L'.
```

Closest Pair of Points

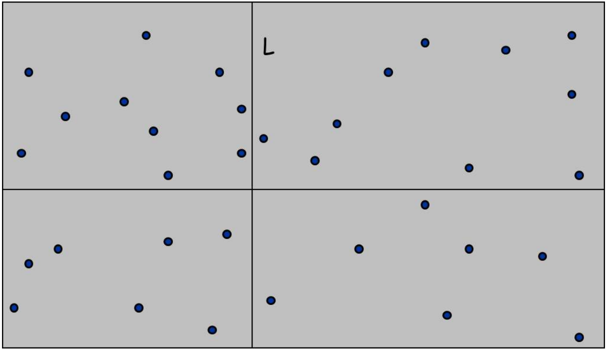
- *Closest pair problem*. Given n points in the plane, find a pair of points with the smallest Euclidean distance between them.
- Fundamental geometric primitive.

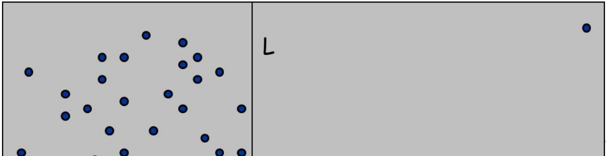
  Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.

  Special case of nearest neighbor, Euclidean MST.
- *Brute force*. Check all pairs of points with $\Theta(n^2)$ comparisons.
- *1-D version*. $O(n \log n)$ if points are on a line.
- *Assumption*. No two points have same x coordinate.

Closest Pair of Points: First Attempt

- *Divide*. Sub-divide region into 4 quadrants.
- *Obstacle*. Impossible to ensure $n/4$ points in each piece.





Homework

- Read Chapter 5 of the textbook.
- Exercises 1 & 2 in Chapter 5.