CS 401

Divide and Conquer

Xiaorui Sun

Divide and Conquer

Divide: We reduce a problem to several subproblems.

Typically, each sub-problem is at most a constant $c < 1$ fraction of the size of the original problem

Conquer: Recursively solve each subproblem

Combine: Merge the solutions

Example:

- Mergesort

Counting inversions

Counting Inversions

Music site tries to match your song preferences with others.

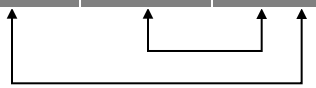
- You rank n songs.
- Music site consults database to find people with similar tastes.

Similarity metric: number of inversions between two rankings.

- My rank: 1, 2, …, n.
- Your rank: $a_1, a_2, \ldots, a_n$.
- Songs i and j inverted if $i < j$, but $a_i > a_j$.

Songs

Me	1	2	3	4	5
You	1	3	4	2	5



Inversions

3-2, 4-2

Brute force: check all $\Theta(n^2)$ pairs i and j.

Counting Inversions: Divide-and-Conquer

Divide-and-conquer.

- Divide: separate list into two pieces.
- Conquer: recursively count inversions in each half.
- Combine: count inversions where a_i and a_j are in different halves, and return sum of three quantities.

Divide: $O(1)$.

Conquer: $2T(n / 2)$

5 blue-blue inversions

8 green-green inversions

9 blue-green inversions

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

Combine: ???

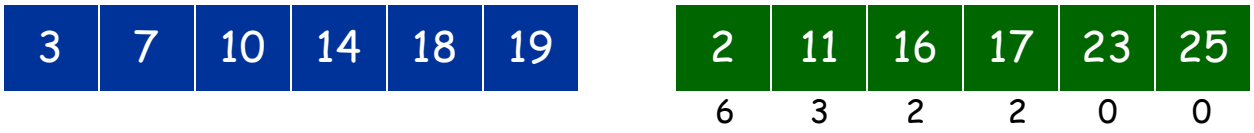
Total = 5 + 8 + 9 = 22.

Enumerate all blue-green pairs takes $O(n^2)$ time

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.



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

Count: O(n)

Combine:

- Sort two halves.
- Count inversions where a_i and a_j are in different halves.

$O(n(\log n)^2)$ time

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.

to maintain sorted invariant

6 3 2 2 0 0

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

Count: $O(n)$

Merge: $O(n)$

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

Counting Inversions: Implementation

Pre-condition. [Merge-and-Count] A and B are sorted.

Post-condition. [Sort-and-Count] L is sorted.

```
Sort-and-Count(L) {
   if list L has one element
      return 0 and the list L

   Divide the list into two halves A and B
   (r_A, A) ← Sort-and-Count(A)
   (r_B, B) ← Sort-and-Count(B)
   (r, L) ← Merge-and-Count(A, B)

   return r = r_A + r_B + r and the sorted list L
}
```

Lesson

Sometimes, it is useful to redefine the problem to make the recursion work

In the counting inversions problem

- The combine step becomes easier if two halves are sorted
- So, we redefine the problem (as well as the subproblems) as finding the number of inversions and sorting the input

Master Theorem

Master Theorem

Suppose $T(n) = a\,T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$. Then,

c: absolute constant

- If $a < b^k$ then $T(n) = \Theta\left(n^k\right)$
- If $a = b^k$ then $T(n) = \Theta\left(n^k \log n\right)$
- If $a > b^k$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

Works even if it is $\left\lceil \frac{n}{b} \right\rceil$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1, k \geq 0$ and $T(n) = O(1)$ for $n \leq b$.

Question

Consider the following recurrence. Which case of the master theorem?

$$T(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ T\left(\left\lfloor \frac{n}{5} \right\rfloor\right) + T\left(n - 3\left\lceil \frac{n}{10} \right\rceil\right) + \frac{11}{5}n & \text{if } n > 1 \end{cases}$$

- A. $T(n) = \Theta(n)$
- B. $T(n) = \Theta(n \log n)$
- C. $T(n) = \Theta(n^2)$
- D. Master theorem not applicable

Master Theorem

Suppose $T(n) = a\,T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$.

$$T(n) = \begin{cases} \Theta(n^k) & \text{if } a < b^k \\ \Theta(n^k \log n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$

Akra–Bazzi theorem

Wiki!

How to use master theorem?

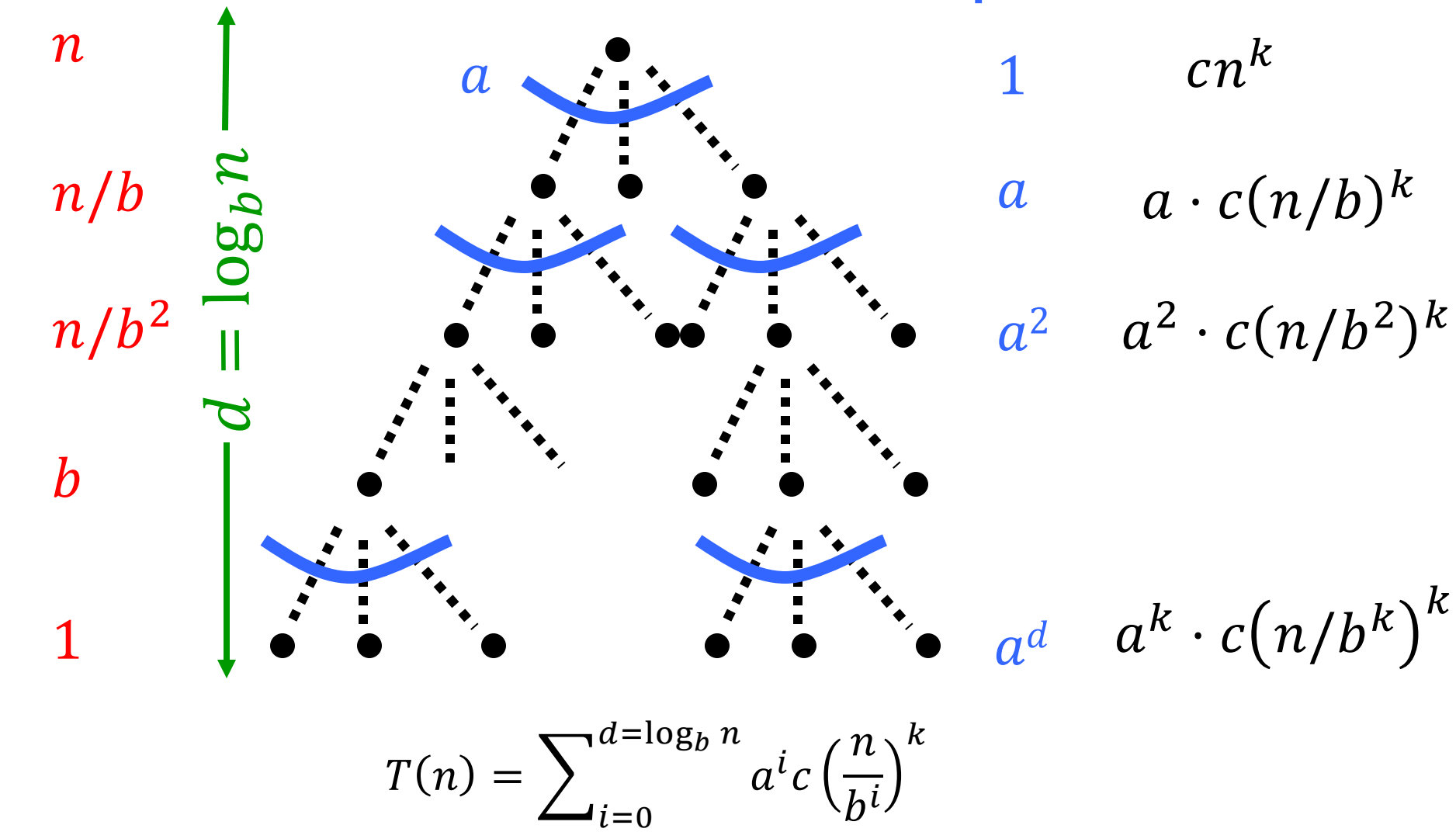
For a divide and conquer algorithm

- a: number of subproblems
- b: ratio of problem size / subproblem size
- $c \cdot n^k$: running time of divide and combine step

We have recurrence $T(n) = a\,T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$.

Example: Mergesort have two subproblems of half size of the original problem, and the cost of divide and combine step is O(n), so T(n) = 2 T(n / 2) + c n, which implies T(n) = Θ(n log n)

Understand Master Theorem



$$T(n) = \sum_{i=0}^{d=\log_b n} a^i c \left(\frac{n}{b^i}\right)^k$$

Understand Master Theorem

Suppose $T(n) = a\,T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$. Then,

- If $a < b^k$ then $T(n) = \Theta\left(n^k\right)$
- If $a = b^k$ then $T(n) = \Theta\left(n^k \log n\right)$
- If $a > b^k$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

# of problems increases slower than the decreases of cost. Top term dominates.

# of problems increases faster than the decreases of cost Bottom term dominates.

$$T(n) = \sum_{i=0}^{d=\log_b n} a^i c \left(\frac{n}{b^i}\right)^k$$