DIVIDE-AND-CONQUER ALGORITHMS, PART 1: INTRODUCTION AND EXAMPLES

Marcus Ritt

CMP 601 – Algorithms and Theory of Computation —

<2020-04-14 ter>

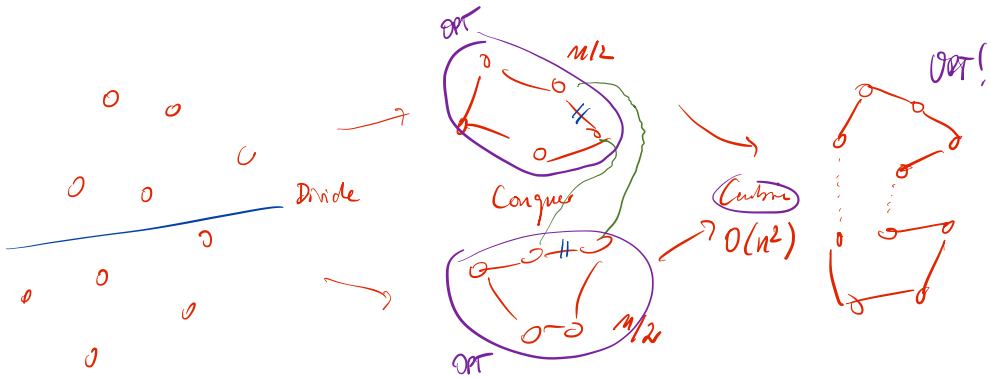
Outline

- Instance I of size n

```
DC(I):=
  if n ≤ n_0 then
    return direct solution of base case
  else
    divide I into subproblems I_1,...,I_k.
    solve s_i = DC(I_i), i ∈ [k] recursively.
    solve I by combining solutions s_1,...,s_k.
  end if
```

- Natural correctness proof is by (complete) induction
- Base: Show that the base case ($n \leq n_0$) is solved correctly
- Step:
  - Induction hypothesis: the algorithm works correctly for smaller instances (size $< n$).
  - Show: it works for instances of size n.
- Main problem here: correctness of the combine step

- Here's a divide-and-conquer algorithm to solve the TSP:
  - Divide the cities into two subsets of the same size.
  - Find the optimal routes for each subset.
  - Combine the two routes optimally, but checking all ways of removing an edge from each, and joining them.
- Recurrence: $T(n) = 2T(n/2) + O(n^2) = O(n^2 \log n)$.
- Correctness?



- Assume: divide takes time $d(n)$, combine $s(n)$, subproblems have size n_i, $i \in [k]$
- We have the *natural recurrence*

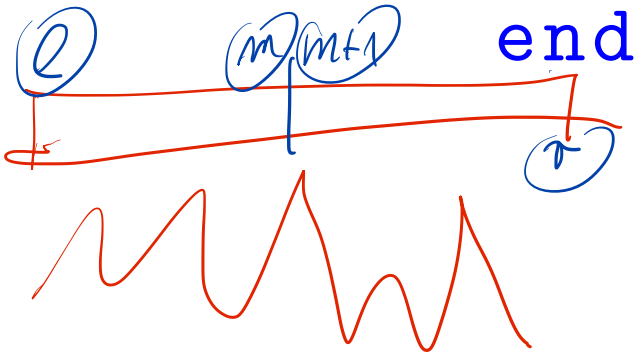
$$T(n) = \begin{cases} \Theta(1), & \text{for } n \leq n_0, \\ \sum_{i \in [k]} T(n_i) + f(n), & \text{otherwise,} \end{cases}$$

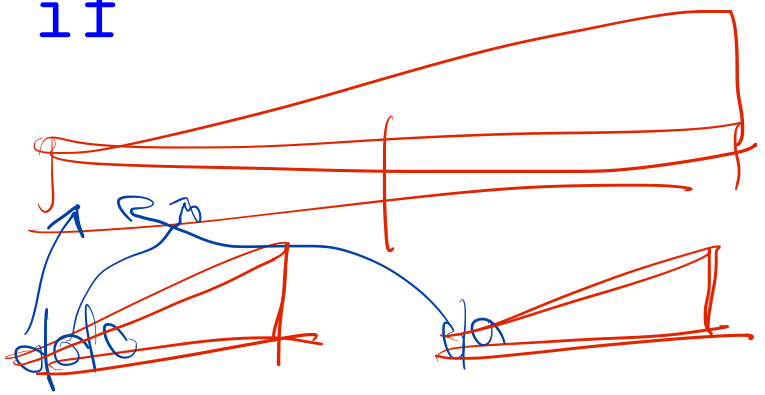
with $f(n) = d(n) + c(n)$. If the subproblems are balanced, this simplifies to

$$T(n) = \begin{cases} \Theta(1), & \text{for } n \leq n_0, \\ kT(\lceil n/k \rceil) + f(n), & \text{otherwise.} \end{cases}$$

(Empty)

```
MergeSort(a, l, r) :=
    if l < r then
        m := ⌊(l + r)/2⌋
        MergeSort(a, m, q)
        MergeSort(a, m + 1, r)
        Merge(a, l, m, r)
    end if
end
```





- Correctness: follows from correctness of "Merge".
- Complexity: $T(n) = 2T(n/2) + O(n) = O(n \log n)$.

$\Theta(n^2)$

Take two n-bit numbers p, q, separate them:

$$p = \boxed{p_l}\boxed{p_r} = p_l\, 2^{n/2} + p_r$$

$$q = \boxed{q_l}\boxed{q_r} = q_l\, 2^{n/2} + q_r.$$

Thus we have:

$$\begin{aligned} pq &= (p_l\, 2^{n/2} + p_r)(q_l\, 2^{n/2} + q_r) \\ &= 2^n p_l q_l + 2^{n/2}\,(p_l q_r + p_r q_l) + p_r q_r. \end{aligned}$$

① ② ③ ④

① $n/2 \times n/2$ ② $n/2 \times n/2$ ③ $n/2 \times n/2$ ④ $n/2 \times n/2$

$T(n) = 4T(n/2) + O(n) = \Theta(n^2)$.

Note:

$$p_l q_r + p_r q_l = (p_l + p_r)(q_l + q_r) - p_l q_l - p_r q_r.$$

1962

$T(n) = 3T(n/2) + O(n)$.

Binary multiplication 2

```
if n = 1 then return pq
else
    x_1 := MULT-BIN(p_l, q_l)
    x_2 := MULT-BIN(p_r, q_r)
    x_3 := MULT-BIN(p_l + p_r, q_l + q_r)
    return x_1 2^n + (x_3 − x_2 − x_1) 2^{n/2} + x_2
end if
```

- Thus: we get away with only three recursive calls
- $T(n) = 3T(n/2) + O(n)$ $= O(n^{1.59})$

$\log_2 3 \approx 1.59$

Simplified recurrences

$$T(n) = \begin{cases} O(1), & n \le n_0 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) & n \ge 2 \end{cases} + O(n),$$

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

- We usually don't write the base case.
- We drop "small perturbations" such as floor and ceilings
- This can be rigorously justified

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

$T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{\log_2 3})$

- (1) Substitution: guess the solution, and prove by induction
- (2) Expansion: expand recurrence tree, sum costs over levels, then levels
- (3) Master Theorem
- Akra-Bazzi's theorem ⟸ NEW outside of KLT.

Guess: $T(n) \leq cn\log_2 n - d \Rightarrow O(n \log n)$.

- Take MergeSort $T(n) = 2T(n/2) + O(n)$

$\hookrightarrow \leq c'n$ given

$T(1) \leq c \cdot 1 \cdot \log_2 1$

$n_0 = 2 \; (= 1)$

$T(2) \leq c \cdot 2 \log_2 2 = c \cdot 2$

(1) $T(n) \leq 2T(n/2) + c'n$

(2) I.H. $\rightarrow \leq 2cn/2 \log_2(n/2) + c'n$

$\leq cn \log_2 n/2 + c'n = cn \log_2 n - cn + c'n$

$\Rightarrow \leq cn \log_2 n$ □

≤ 0 residual.

for $c \geq c'$.

$T(n/2) \leq c\, n/2 \log_2 n/2$

$c' > 0$

$\log_2 n/2 = \log_2 n - \log_2 2$

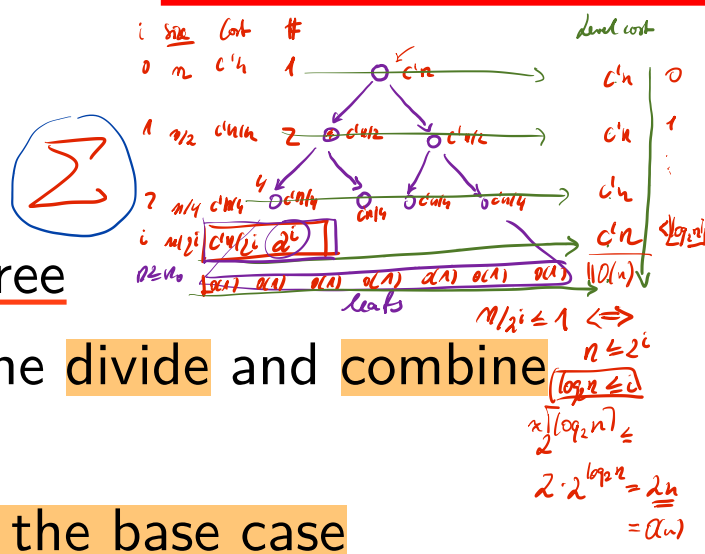
$-cn + c'n \leq 0$

$c' \leq c$

$c < c'$

$c > 0, c' > 0$

$c = c'/2$



- A recursive algorithm corresponds to a tree
- We can label the internal vertices with the divide and combine costs
- We can label the leaves with the cost of the base case
- Thus: summing over all nodes we get the total cost
- Idea: sum over levels, then the levels
- Expansion is easiest for homogenous cases

$$T(n) = aT(bn) + f(n)$$

$a = 2 \quad b = \frac{1}{2}$

$$\sum_{0 \le i < \log_2 n} cn + O(n) =$$

$\log_2 n$

$$O(cn \log_2 n + O(n)) = O(n \log n).$$

Expansion of a homogeneous recurrence 1

- We have a^i subproblems in level i
- The size of the subproblems in level i is $b^i n$
- Thus the cost of each internal node is $f(b^i n)$
- And the total level cost is $a^i f(b^i n)$
- We reach the base case $n \leq n_0$ for $i \geq \overline{\beta} = \log_{1/b} n/n_0$
- So the internal cost is

$$\sum_{0 \leq i < \overline{\beta}} a^i f(b^i n).$$

(Empty)

Expansion of a homogeneous recurrence 2

- The number of leaves is $O(a^{\log_b n}) = O(n^{\log_{1/b} a})$
- They cost $O(1)$, so their total cost is also $O(n^{\log_{1/b} a})$
- So overall we have

$$\sum_{0 \le i < \log_b n} a^i f(b^i n) + O(n^\alpha)$$

- where $\alpha = \log_{1/b} a$.

Example $T(n) = 3T(n/4) + O(n^2)$

Example: Variations of MergeSort

- What happens if you divide the sequence into three parts? Or four?