VE281

Data Structures and Algorithms

Dynamic Programming

Announcement

• Pre-test for programming project three is available online.

- Participate in the online course evaluation "IDEA".
 - It will close on Dec. 16th.
 - Follow the link in an email sent to your SJTU email account.

Review

- Quick Sort
- Comparison Sort Summary and Time Complexity
 - The worst case time complexity for any comparison sort is $\Omega(n \log n)$.
- Non-Comparison Sort
 - Counting Sort and Bucket Sort
 - Radix Sort

Outline

- Motivation of Dynamic Programming
- Example: Matrix-Chain Multiplication
- Summary

Algorithm Design Methods

- We have already learned two ways to design algorithms:
 - Greedy method.
 - Divide and conquer.
- Some more design methods:
 - Dynamic programming.
 - Backtracking.
 - Branch and bound.

Greedy Method

- Solve problem by making a sequence of decisions.
- Decisions are made one by one in some order.
- Each decision is made using a **greedy** criterion.
- A decision, once made, is usually not changed later.
- Example: Dijkstra's algorithm and Prim's algorithm

Divide and Conquer

- Given a problem to be solved, **split** the problem into several, smaller sub-problems (often recursively).
- Solve each sub-problem independently.
- Combine the solutions to the sub-problems to yield a solution to the original problem.
- Examples: merge sort and quick sort.

Limitation of Divide and Conquer

- Recursively solving sub-problems can result in the same computations being repeated when the subproblems overlap.
- For example: computing the **Fibonacci sequence** $f_0 = 0$; $f_1 = 1$; $f_n = f_{n-1} + f_{n-2}, n \ge 2$

• Divide and conquer approach:

```
int fib(int n) {
  if(n <= 1) return n;
  return fib(n-1)+fib(n-2);
}</pre>
```

Fibonacci Sequence

Divide and Conquer Solution

```
int fib(int n) {
   if(n <= 1) return n;</pre>
   return fib(n-1)+fib(n-2);
                              fib(5)
                                            fib(3)
                fib(4)
                                       fib(2) fib(1)
                     fib(2)
        fib(3)
   fib(2) fib(1) fib(1) fib(0) fib(1) fib(0)
                   Subproblems are overlapped. Much computation
fib(1)fib(0)
                   is wasted. Time complexity is \Omega(1.5^n).
```

Fibonacci Sequence Iterative Solution

• We can also compute the Fibonacci sequence in iterative way:

```
int fib(int n) {
  f[0] = 0; f[1] = 1;
  for(i = 2 to n)
    f[i] = f[i-1]+f[i-2];
  return f[n];
}
```

• Time complexity is $\Theta(n)$.

Dynamic Programming

- Used when a problem can be divided into subproblems that overlap.
 - Solve each sub-problem once and store the solution in a table.
 - ⇒ Trading space for time.
 - If a sub-problem is encountered again, simply look up its solution in the table.
 - Reconstruct the solution to the original problem from the solutions to the subproblems.
- The more overlap the better, as this reduces the number of sub-problems.
- Dynamic programming can be applied to solve optimization problem.

Optimization Problem

- Many problems we encounter are optimization problems:
 - A problem in which some function (called the objective function) is to be optimized (usually minimized or maximized) subject to some constraints.
- The solutions that satisfy the constraints are called **feasible solutions**.
- The number of feasible solutions is typically very large.
- We obtain the optimal solution by **searching** the feasible solution space.

Optimization Problem Example

- Minimum spanning tree.
 - Constraints: the subgraph must be a spanning tree.
 - Objective function: the sum of all edge weights.

Outline

- Motivation of Dynamic Programming
- Example: Matrix-Chain Multiplication
- Summary

- What is the cost of multiplying two matrices A and B?
 - Suppose A is a $p \times q$ matrix and B is a $q \times r$ matrix.
 - Since the time to compute C = AB is dominated by the number of scalar multiplications, we use the number of scalar multiplications as the complexity measure.
- $\bullet \ C_{ij} = \sum_{k=1}^q A_{ik} B_{kj}.$
 - We need q scalar multiplications to calculate C_{ij} .
 - C is of size $p \times r$.
- The number of scalar multiplications is pqr.

- Now how would you compute the multiplication of three matrices $A \times B \times C$?
 - Suppose A is of size 100×1 , B is of size 1×100 , and C is of size 100×1 .
- If we multiply as $(A \times B) \times C$, the number of scalar multiplications is 20000.
- If we multiply as $A \times (B \times C)$, the number of scalar multiplications is 200.

- If we want to multiply a chain of matrices $A_1 \times A_2 \times \cdots \times A_n$, where A_i is of size $p_{i-1} \times p_i$, what is the best order of multiplication to minimize the number of scalar multiplications?
- This is an optimization problem.
- How many different orders on matrix multiplication?

Matrix-Chain Multiplication Number of Orders

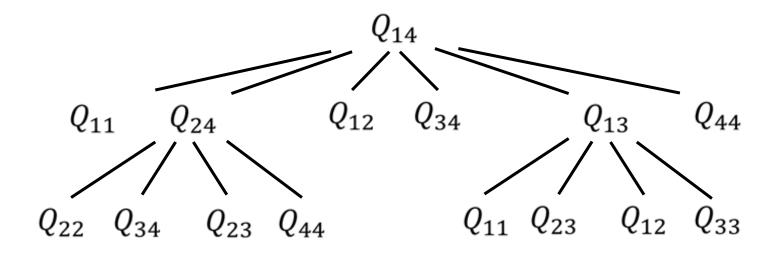
- Suppose the number of order is P(n) for multiplying n matrices.
- Suppose the last multiplication is $(A_1 \times \cdots \times A_k) \times (A_{k+1} \times \cdots \times A_n)$.
 - The number of possible order is P(k)P(n-k).
- Thus $P(n) = \sum_{k=1}^{n-1} P(k) P(n-k)$.
- It can be proved that $P(n) = \Omega(4^n/n^{1.5})$.
- Instead of enumerating all of the orders, can we do better to solve the optimization problem?

- For simplicity, define the problem of finding the optimal order to multiply $A_i \times A_{i+1} \times \cdots \times A_j$ as Q_{ij} . The minimal number of scalar multiplication is m_{ij} .
 - We ultimately want to solve Q_{1n} .

- Suppose in the optimal order for $A_i \times \cdots \times A_j$, the last multiplication is $(A_i \times \cdots \times A_k) \times (A_{k+1} \times \cdots \times A_j)$.
- Then the order of computing $A_i \times \cdots \times A_k$ in the optimal order of computing $A_i \times \cdots \times A_j$ must be an optimal order to compute $A_i \times \cdots \times A_k$.
 - Why?
 - If not, we have a better order for computing $A_i \times \cdots \times A_j$.
 - Similar conclusion for computing $A_{k+1} \times \cdots \times A_j$.
- If we know k, we can divide the problem Q_{ij} into two smaller instances: Q_{ik} and $Q_{(k+1)j}$.

- Assume we have known the minimum number of scalar multiplications for Q_{ik} and $Q_{(k+1)j}$ as m_{ik} and $m_{(k+1)j}$.
 - Then $m_{ij} = m_{ik} + m_{(k+1)j} + p_{i-1}p_kp_j$.
- However, we don't know k. We need to consider all possible divisions, i.e., all $i \le k \le j-1$.
- Thus, in order to solve Q_{ij} , we need to consider all subproblems Q_{ik} and $Q_{(k+1)j}$, for all $i \le k \le j-1$.
 - $m_{ij} = \min_{i \le k \le j-1} (m_{ik} + m_{(k+1)j} + p_{i-1}p_k p_j)$

• In summary, we can divide the problem into subproblems of the same form.



Many subproblems are overlapped.

- The straightforward recursive algorithm has exponential time complexity.
 - However, it will encounter each subproblem many times in different branches of the tree.
- The total number of different subproblems is not exponential.
 - They are Q_{ij} , for $1 \le i \le j \le n$.
 - The total number is n(n+1)/2.
- Instead, we use a tabular, bottom-up approach.

Matrix-Chain Multiplication Bottom-up Approach

- Apply the recursive relation: $m_{ij} = \min_{1 \le k \le j-1} (m_{ik} + m_{(k+1)j} + p_{i-1}p_kp_j)$
- Initial situation $m_{11} = m_{22} = \dots = m_{nn} = 0$.
- In the first round, we compute m_{12} , m_{23} , ..., $m_{(n-1)n}$.
- In the second round, we compute m_{13} , m_{24} , ..., $m_{(n-2)n}$.
- So on and so forth. In the l-th round, we compute $m_{1(l+1)}, m_{2(l+2)}, \dots, m_{(n-l)n}$.
- Finally, we compute m_{1n} .
- We also record the partition k which gives the minimal m_{ij} in s_{ij} .

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | 1 | enteredients to the second sec | a par a par par bases | 4 |
|--|---|--|--------------------------------|---|
| 1 | 0 | | | |
| 1: So portakas anjamin kipajajali Norami Mariji M. Limoral Sali Alia Mariji M. Nora, najah Orienger (Alia Mariji M. | _ | 0 | | |
| Take an property (III) sins This are of the exp. | _ | _ | 0 | |
| 4 | _ | _ | _ | 0 |

| | T(LeftSz) | 1 | 1 Contractive to the second of | a par a par par bases | 4 |
|---|-----------|---|--|--------------------------------|---|
| | 1 | | | | |
| One perchase, regiments bytasquits any the research of the condition of the condit | 2 | _ | _ | | |
| Hadrationpose. The copiety of \$1, size \$1 desired to any. | 3 | _ | _ | _ | |
| | 4 | | | | _ |

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | 1 | Security of the security of th | 8 pm 8 apr | 4 |
|--|---|--|-------------------|---|
| 1 | 0 | | | |
| to to praction asymmetric hypothysis on the association of the asymmetric distribution of the | _ | 0 | | |
| Take some green. Take s | _ | _ | 0 | |
| 4 | _ | _ | _ | 0 |

| | T(LeftSz) | 1 | statement is at a series of the series of th | 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - | 4 |
|--|-----------|---|--|---|---|
| | 1 | | | | |
| : do piciales segiondo leptopilo es; la transcial signé di la la transcial signé di la transcial signé di dimiti (leptot. è la man park si dimiti (leptot. è ES para El picial si dimiti si con di la considera di la disconsidera di la disconsi | 2 | | | | |
| Facilitation pair. The complete ARE, size Exclusively any. | 3 | _ | _ | _ | |
| | 4 | _ | _ | _ | _ |

T(RightSz)

Recursive relation:
 TONO = TONOCCOO + TONOCCOO + BOOCH

Worst case happens when each time the prior is the smallest firm or the largest form.
 T(N) = T(N - 1) + T(0) + B(N)

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| | $\Theta(N)$ | | The second of th | 1 40° | |
|---|-------------|---|--|-------|-----|
| | | 1 | 2 | 3 | 4 |
| | 1 | 0 | 100 | | |
| On pickiden, registede liptojelde en; her namel allydell, herant i allveld (medil) (rejekt, herant janlveld (medil) (rejekt, filozopa) (vidalijani (hidana)); Oberen, mjeli de limbanji (hidana); | 2 | | 0 | 10 | |
| Sanda mitrarry no. Tar-complete; in REF, when Fix the six of the area. | 3 | | | 0 | 200 |
| | 4 | | _ | _ | 0 |

| | T(LeftSz) | | extramerism is at all. A state of the control of the control of the control of the | | | | |
|---|-----------|---|---|---|---|--|--|
| | (-)) | 1 | 2 | 3 | 4 | | |
| | 1 | _ | 1 | | | | |
| the periodnes, registered beginning bla- ren. The research of ple 4. The research of the feet of the period of ple 4. The research of the feet of the period of ple 4. The research of the feet of the period of the feet of the period of the feet of the period of the period of the feet of the period of the p | 2 | | | 2 | | | |
| andra militarije ma. Na completelj in NEU, sleze K in Stevieralda proj. | 3 | _ | _ | _ | 3 | | |
| | 4 | _ | _ | _ | _ | | |

T(RightSz)

Recursive relation:
 TONO = TONOCCOO + TONOCCOO + BOOCH

Worst case happens when each time the prior is the smallest item or the largest form.
 T(N) = T(N-1) + T(0) + B(N)

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | 1 | consecutive to the second control of the sec | _ | 4 | T(LeftSz) | 1 | enception of the second of the | - | 4 |
|--|---|--|----|-----|---|---|--|---|---|
| 1 | 0 | 100 | | | 1 | _ | 1 | | |
| . To pricious argument deplaquido en construir de place la la transmit de al place la la transmit de al place la la transmit de al place la la la transmit de place la la la la place la la la la place la | _ | 0 | 10 | | 1. In previous regression departs de on 1. Ser conservative de 1. Servative des conservatives de 1. Servative des conservatives de 1. Servative de conservative de 1. Servative de 1. Servative 1. Servative de 1. Servative 1. Servative 1. Servative | | | 2 | |
| *technologies. *In complete, 45%, des 2 de des cille ares | _ | _ | 0 | 200 | taka era ese. *Images (All darla kanila esp. | | | | 3 |
| 4 | | | | 0 | 4 | | | | |

$$= T(N-1) + T(0) + dN$$
$$= T(N-2) + 2T(0) + d(N-1) + dN$$

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| | $\Theta(N)$ | 1 | the manufact is at the control of th | 14 pm 14 pm 15 pm 16 pm 16 pm | 4 |
|--|-------------|---|--|---|-----|
| | 1 | 0 | 100 | | |
| Do peri hor, registrele lipskylde en Do peri hor, registrele lipskylde en Do same Lief lipskylde Do same Lief lipskylde | 2 | _ | 0 | 10 | |
| + Sauda mitoraryona. • The complete HEE, since The desired leaves. | 3 | _ | | 0 | 200 |
| | 4 | _ | _ | _ | 0 |

| | T(LeftSz) | 1 | Section 1. | age de de | 4 |
|---|-----------|---|---|-----------------|---|
| | 1 | _ | 1 | | |
| the periodes augments lightly life on the control of 1964 the control of 1964 the control of the signifiest the control of the signifiest the control of the signifiest the control of the significant the control of the significant the control of the significant the significant the significant the significant | 2 | | _ | 2 | |
| Faulustrospon. *The copiety of EL des Extrincials any | 3 | | _ | _ | 3 |
| | 4 | | _ | _ | _ |

$$= T(0) + NT(0) + d + 2d + \cdots + d(N-1) + dN$$

$$= min\{20, 100\}$$

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| | $\Theta(N)$ | 1 | the manufact is at the control of th | 14 pm 14 pm 15 pm 16 pm 16 pm | 4 |
|---|-------------|---|--|---|-----|
| | 1 | 0 | 100 | 20 | |
| (the periodnes required in legal politically in the control of political in the control of the | 2 | | 0 | 10 | |
| Sada enterque. • Tar complete a RIS, den Riche involle pre- | 3 | | | 0 | 200 |
| | 4 | _ | | _ | 0 |

| | T(LeftSz) | 1 | Security of the second of the | a ya a ugo ⇔ an | 4 |
|---|-----------|---|---|--------------------------|---|
| | 1 | | 1 | 1 | |
| : the periodes required a liquid plane, any | 2 | | | 2 | |
| Hands entrançais. *The complete HEE, since Facilities delt any. | 3 | | | | 3 |
| | 4 | | | | |

$$= T(0) + NT(0) + d + 2d + \cdots + d(N-1) + dN$$

$$-$$
 = min{20, 100}

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | | of legal of the control of the contr | 1 pm 1 pm 1 pm 1 pm 1 pm 1 pm | |
|--|---|--|--|-----|
| | 1 | 2 | 3 | 4 |
| 1 | 0 | 100 | 20 | |
| Two price loss, regiments frequents on the contract of the con | | 0 | 10 | |
| untre acrepa. Secreptia + 80, deal Tu to octob any. | | _ | 0 | 200 |
| 4 | | _ | _ | 0 |

| T(Lef | <u>tSz)</u> | to another to the stage of the | The person of th | 4 |
|--|-------------|---|--|---|
| 1 | | 1 | 1 | |
| 1. Stoperishes, equivals lepasjulle equi la levenur (signification segment | _ | _ | 2 | |
| Tale management the complex of the first translations. | _ | _ | _ | 3 |
| 4 | _ | _ | _ | _ |



 $T(N) = T(LeftSs) + T(R(ghtSs) + \Theta(N)$

Average case time complexity of quick sort can be proved to \(\theta(N)\) log N3.

Best case happens when each time the pivot divides the array into two equal-sized ones, $= T(N) = T(cN-1)/23 + T(cN-1)/23 + \Theta(N)$ * The recursive relation is similar to that of merge sort.
* $T(N) = \Theta(N) \log N$

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | 1 | The second secon | 1 pm 1 pm 1 pm 1 pm 1 pm 1 pm | 4 |
|--|---|--|--|-----|
| 1 | 0 | 100 | 20 | |
| ho perchase approvide spanyale on the Namaron Cariffel Carrows and Cariffel Carrows and Cariffel Carrows after Administration for the Namaron Cariffel Carrows after Administration for Expressive School Cariffel | _ | 0 | 10 | 30 |
| Stark en en en en. Te rempiris till, des Tick en elle en | _ | _ | 0 | 200 |
| 4 | _ | _ | | 0 |

| | T(LeftSz) | 1 | 1 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1 | a par a par par bases | 4 |
|--|-----------|---|--|--------------------------------|---|
| | 1 | _ | 1 | 1 | |
| : Occ perchas, regiments legalquits erg. In terrate of the limit of th | 2 | | | 2 | 3 |
| Fauls attemptes. *Taxonylety (EE) size Extrincials any. | 3 | | | | 3 |
| | 4 | _ | _ | _ | _ |



 $T(N) = T(LeftSs) + T(R(ghtSs) + \Theta(N)$

Average case time complexity of quick sort can be pro-to O(N log N).

Best case happens when each time the pivot divides the array into two equal-sized ones, $= T(N) = T(cN-1)/23 + T(cN-1)/23 + \Theta(N)$ * The recursive relation is similar to that of merge sort.
* $T(N) = \Theta(N) \log N$

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| Θ(| N) | 1 | management in the stage of the | 1 pm 1 pm 1 pm 1 pm 1 pm 1 pm | 4 |
|--|----|---|---|--|-----|
| 1 | | 0 | 100 | 20 | |
| . the precision, required depailed on the precision of th | | | 0 | 10 | 30 |
| National participate ART, deal for deal for any | | | | 0 | 200 |
| 4 | | _ | _ | | 0 |

| T(LeftSz | 1 | Section 1 and 1 an | te pre n. opti nic | 4 |
|--|---|--|--------------------------|---|
| 1 | _ | 1 | 1 | |
| 1. the perchalan, regionals leptopole reg. 1. learned leaf 1944. 1. learned leaf 1940. 1. learned lea | _ | | 2 | 3 |
| Table and the specific of the transmission of the specific of | _ | _ | _ | 3 |
| 4 | _ | _ | _ | _ |

- Tofact established

In-place
 In-place partitioning
 Worst case needs OCO stack space
 Average case needs OCO NO stack space

 $n = 4, A_1 \times A_2 \times A_3 \times A_4.$

• $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| | $\Theta(N)$ | 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | i gen or or | 4 |
|---|-------------|---|---------------------------------------|-------------------|-----|
| | 1 | 0 | 100 | 20 | |
| : this perceives required in legal pills and the control of pill to the control of pill to the control of pill to the control of the control | 2 | _ | 0 | 10 | 30 |
| Hands entergrass. *The complete ASE, who Excit and Lang. | 3 | _ | | 0 | 200 |
| | 4 | _ | | _ | 0 |

| | T(LeftSz) | Microsoftwish is set in a set of the control of th | | | | |
|--|-----------|--|---|---|---|--|
| ļ | (-)) | 1 | 2 | 3 | 4 | |
| | 1 | | 1 | 1 | | |
| no for idea, maginerale legale; ble o, or control of all \$4.1 more facilities less (\$1) report control of less ideas(\$1) report or, rep \$10 less ideas (\$1) or, rep \$10 less ideas (\$1) | 2 | | | 2 | 3 | |
| ele estruny eus. completé, e 1815, elec 18 de circite de ens, | 3 | _ | _ | _ | 3 | |
| | 4 | | _ | _ | _ | |

| | Werst Case Time | Case Time | In Place | Stable |
|------------|--------------------|---------------|----------|--|
| Insertion | 0 C N 2 3 | 0 C N 2 3 | Year | Year |
| Selection | O(N2) | 0 CN 23 | Year | ~~ |
| Districts | 0 (N ×) | 0(N2) | ***CM | ************************************** |
| Heap Sort | OCN log NO | OCN log NO | 200 | 1710 |
| Morge Sort | OCN log NO | OCN log NO | No | "Y'com |
| Quiek Sort | | S.S.C63886082 | | No |

Insertion sort corrects one reverse-ordered pair at a time
 Quick sort moves elements far distances, correcting multiple

Why is quick mort's worst-case O(N°) while merge sort has the choice of pivot determines size of partitions in quick sort.

- $n = 4, A_1 \times A_2 \times A_3 \times A_4.$
- $p_0 = 10, p_1 = 1, p_2 = 10, p_3 = 1, p_4 = 20.$

| $\Theta(N)$ | 1 | entering to the control of the contr | | 4 | T(LeftSz) | 1 | Section of the sectio | te pre n. open ni- | 4 |
|--|---|--|----------|-----|--|---|--|--------------------------|---|
| 1 | 0 | 100 | 20 (| 220 | Optimal Value 1 | I | 1 | 1 | 3 |
| I the piece has asymon de lepajorito on the asymon de lepajorito on the asymon de lepajorito on the asymon de lepajorito de la terra esta de lepajorito de la terra esta de lepajorito del lepajorito de lepajorito de lepajorito del l | | 0 | 10 | 30 | h produce agrands depuglis or to some designits i to see all consistent con | | | 2 | 3 |
| *shaper (K, dart shader) | | _ | 0 | 200 | *busyandfi darida sadana | | | | 3 |
| 4 | _ | _ | <u> </u> | 0 | 4 | | | _ | |

| | Werst Case | Case Time | In Place | Stable |
|------------|------------|---------------|--|--|
| Insection. | 0 C N 2 3 | 0 CN 2 3 | Year | ************************************** |
| Selection | O(N2) | 0 CN 23 | "Y'mm | ~ |
| Buildide | 0 (N ×) | 0(N2) | ************************************** | 7000 |
| Heap Sort | OCN log NO | OCN log NO | 2000 | 1710 |
| Morgo Sort | OCN log NO | O(N log N) | No | ************************************** |
| Quiek Sort | | S.S.C63886082 | and Market 12 | No |

Insertion sort corrects one reverse-ordered pair at a time
 Quick sort moves elements far distances, correcting multiple

Why is quick sort's worst-case G(N*) while merge sort has
no such a problem?
 The choice of pivot determines size of partitions in quick sort.

Matrix-Chain Multiplication Constructing an Optimal Order

• For comparison sort, is $O(N \log N)$ the best we can do in the worst case?

• Theorem: A sorting algorithm that is based on pairwise comparisons must use $\Omega(N \log N)$ operations to sort in the worst case.

• Proof: Consider the decision tree.

- Decision tree is a binary tree.
- The sorting result is at one of the leaves following the results of a sequence of pairwise comparisons.
- The number of pairwise comparisons in the worst case corresponds to the deepest leaf in the decision tree, or the N
 - 1 height of the tree. 3
- The number 2 fleases in the approximation of the control of the co
- 3 [N!, i.e., the number of permutations on N items. $(A_2 \times A_3)$
- 4 Since a binary tree of height h has the state of height of the decision tree is at least $\lceil \log N! \rceil$.

$$A_1 \times A_2 \times A_3 \times A_4 = (A_1 \times (A_2 \times A_3)) \times A_4$$

Matrix-Chain Multiplication Time Complexity

- **Radix sort** sorts integers by looking at one digit at a time.
- Procedure: Given an array of integers, from the least significant bit (LSB) to the most significant bit (LSB), repeatedly do stable bucket sort according to the current bit.
- For sorting base-*b* numbers, bucket sort needs *b* buckets.
 - For example, for sorting decimal numbers, bucket sort needs 10 buckets.

Matrix-Chain Multiplication Summary

- Matrix-chain multiplication is an optimization problem.
- The solution is based on dynamic programming.
 - The original problem can be divided into same subproblems that **overlap**.
 - Each subproblem is solved once and stored in a table.
 - If a subproblem is encountered again, simply look up its solution in the table.
 - Reconstruct the solution to the original problem from the solutions to the sub-

Outline

- Motivation of Dynamic Programming
- Example: Matrix-Chain Multiplication
- Summary

Dynamic Programming for Optimization

- There are two key ingredients that an optimization problem must have in order for dynamic programming to apply:
 - Optimal substructure;
 - Overlapping subproblems.

Optimal Substructure

- An optimal solution to the problem contains within it optimal solutions to subproblems.
 - In matrix-chain multiplication, the optimal order on calculating $A_i \times \cdots \times A_j$ that splits the product between A_k and A_{k+1} contains within it optimal solutions to the problem of ordering $A_i \times \cdots \times A_k$ and $A_{k+1} \times \cdots \times A_j$.
- You can show optimal substructure property by supposing that each of the subproblem solutions is not optimal and then deriving a contradiction.

Overlapping Subproblems

- A recursive algorithm for the problem solves the same subproblems over and over, rather than always generating new subproblems.
 - E.g., subproblems of matrix-chain multiplication overlap.

- In contrast, a problem for which a divide—and—conquer approach is suitable usually generates brand—new problems at each step of the recursion.
- Dynamic-programming algorithms take advantage of overlapping subproblems by solving each subproblem once and then storing the solution in a table where it can

Designing a Dynamic-Programming Algorithm

- 1. Characterize the structure of an optimal solution.
- 2. Recursively define the value of an optimal solution.
- 3. Compute the value of an optimal solution, typically in a **bottom-up** fashion.
- 4. Construct an optimal solution from computed information.

Memoization

- In dynamic programming, solutions to subproblems are pre-computed and stored in a table.
 - A bottom-up approach.

in the tehle and return it

- An alternative approach is to "memoize" during the recursion.
 - A top-down approach. Start from the large subproblem.
 - When a subproblem is first encountered as the recursive algorithm unfolds, its solution is computed and then stored in a table. Each subsequent time that we encounter this subproblem, we simply look up the value stored

Reference

- Introduction to Algorithms (3rd Edition), by Thomas Cormen et. al., MIT Press (2009)
 - Chapter 15 Dynamic Programming