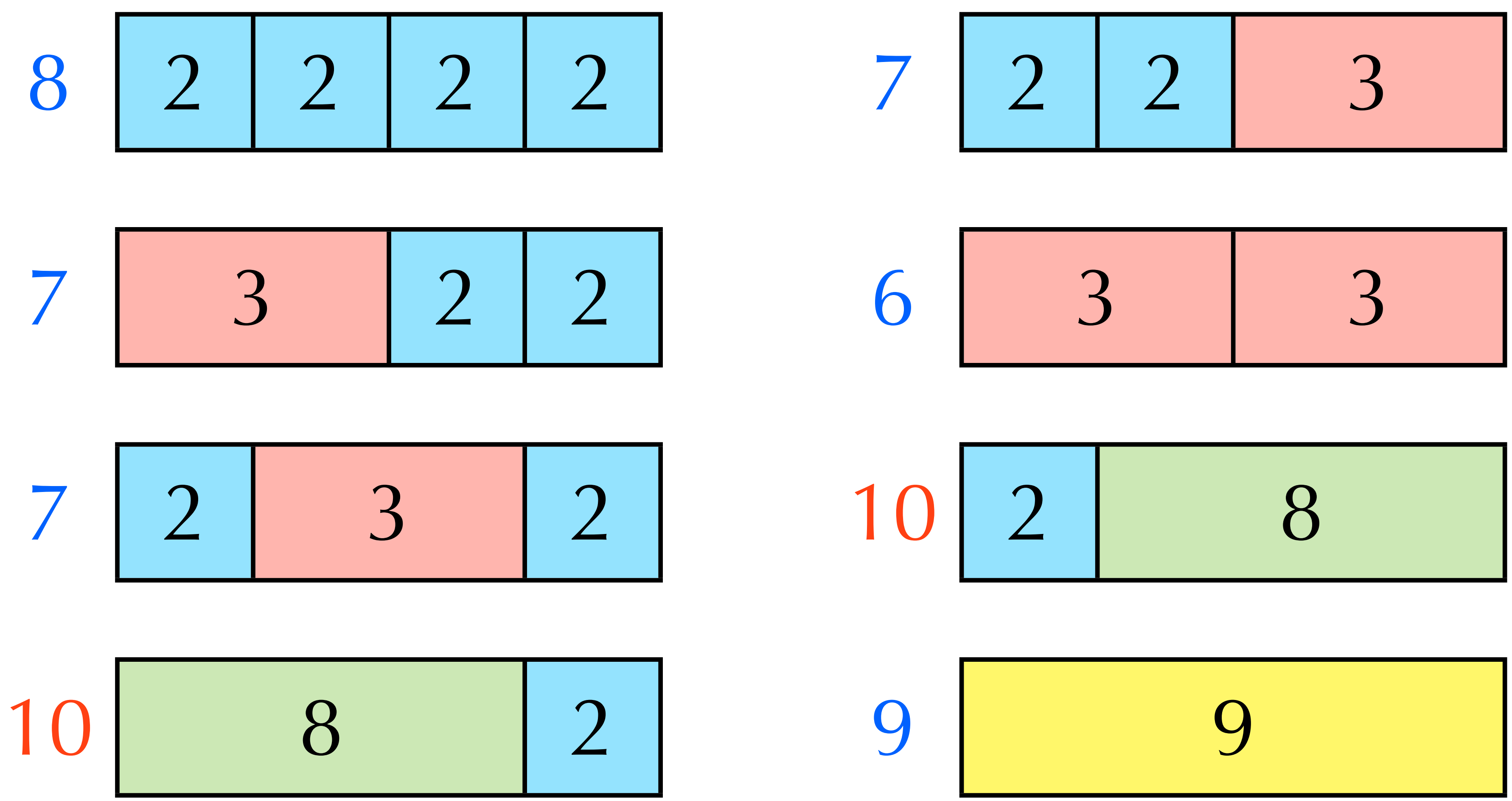
Dynamic Programming

Rod Cutting Problem

- A company buys long steel rods and cut them into shorter rods. Cutting is free
- The company sells each rod of length x for p_x dollars for $x \in \{1,\ldots,k\}$.
- Input: n, k, $p_1,\ldots,p_k$
- Output (easy): the maximum revenue obtainable by cutting a rod of length n and selling the pieces
- Output (hard): optimal cutting

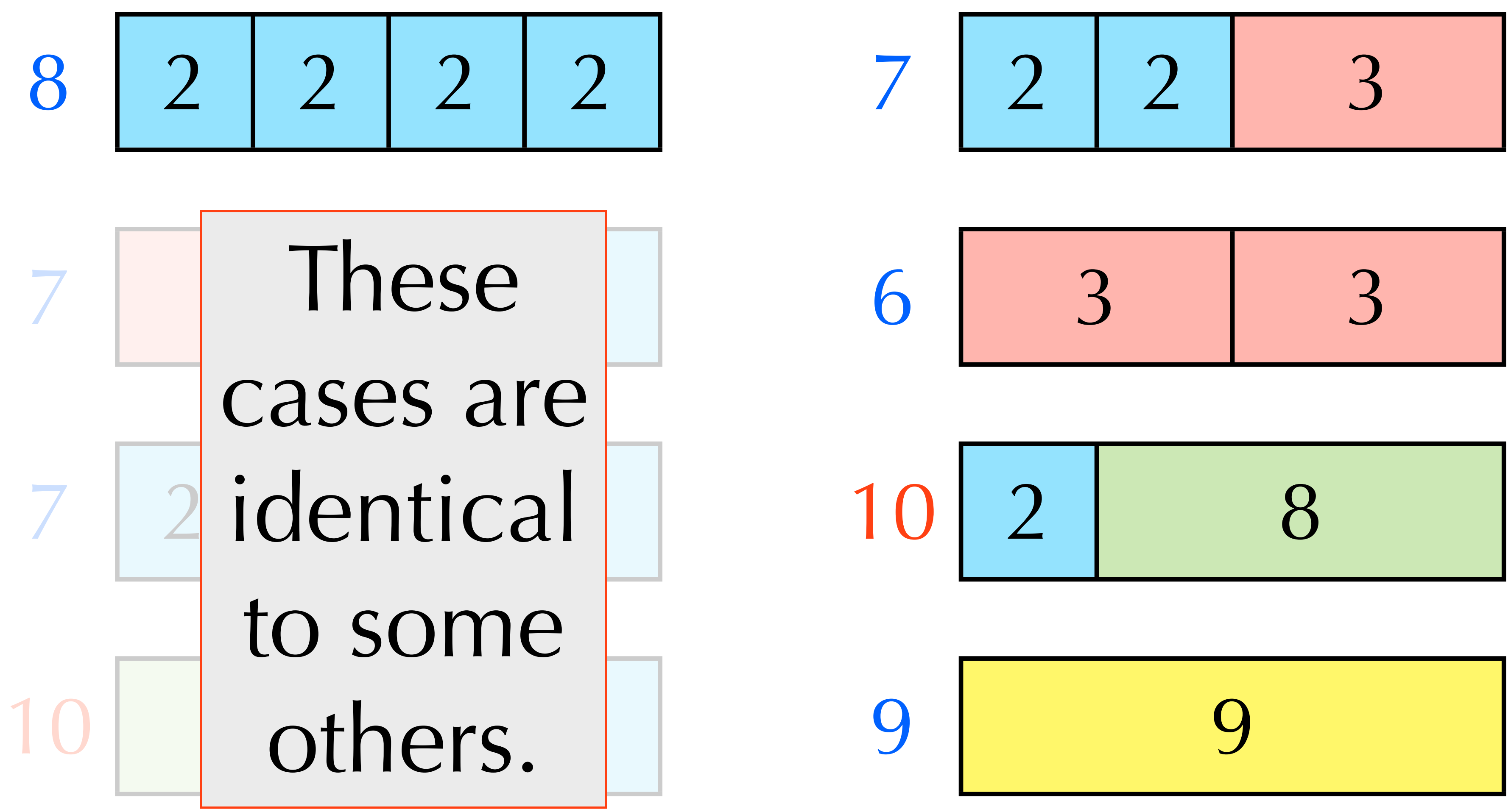
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9



Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9

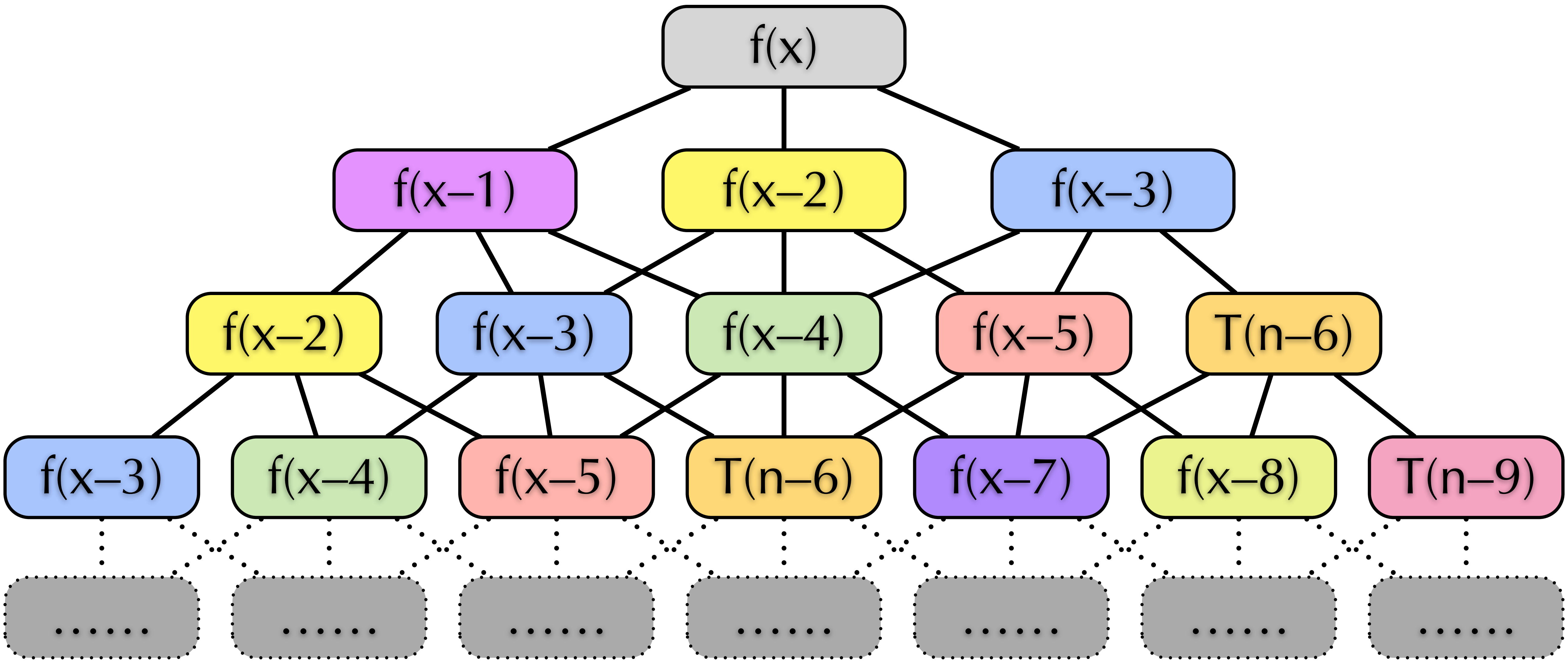


Divide and Conquer

- f(x): the max revenue of rod of length x
- Termination: If x=0, return 0.
- Divide: Let $y_i=f(x-i)$ for $i\in\{1,\ldots,\min(k,x)\}$
  - Let $k'=\min(k,x)$.
- Conquer: Compute y_i
- Combine: return $\max(p_1+y_1,\ldots,p_{k'}+y_{k'})$

$$f(x) = \max_{1\leq i\leq \min(k,x)} \left(f(x-i)+p_i\right)$$

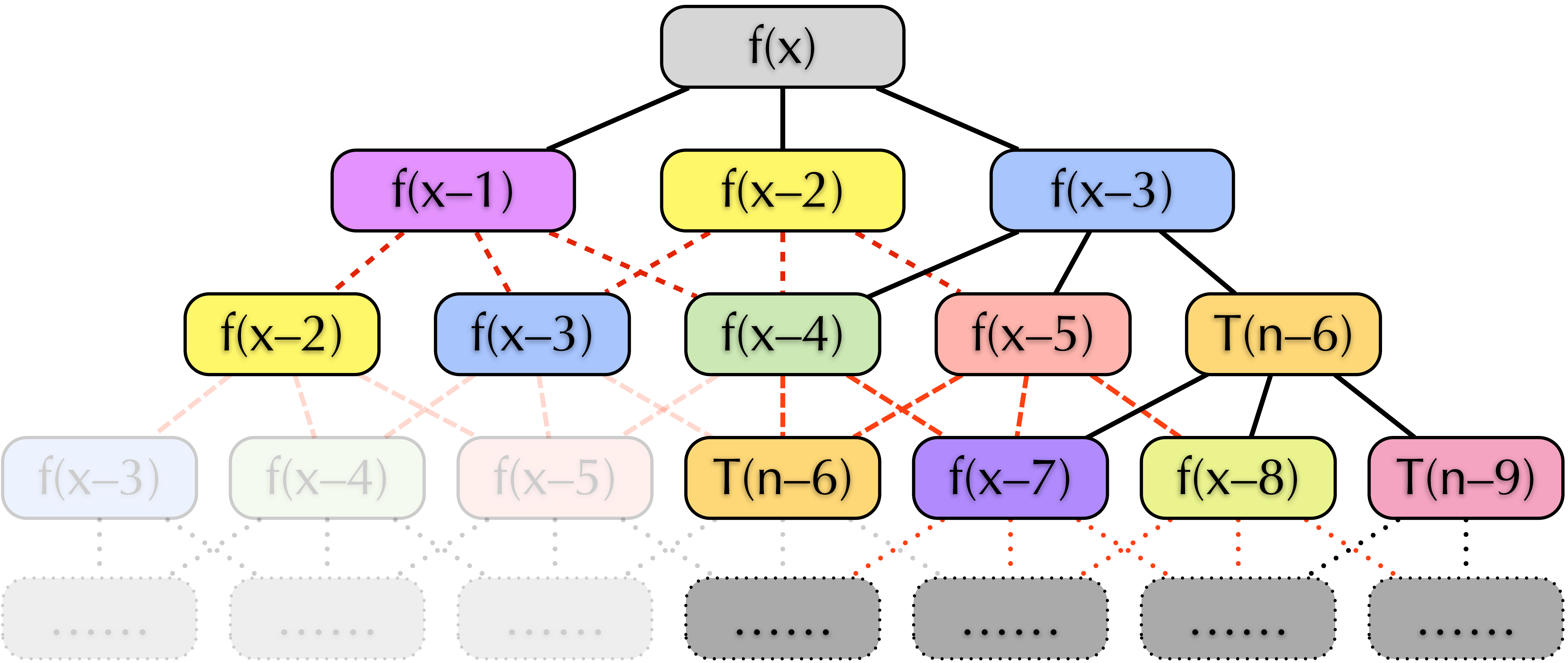
Example: k=3



The problem will be solved in $\Omega(c^n)$-time for some c>1!

Observation: some subproblems are identical!

Example: k=3



If we only compute f(x–i) once for each i...

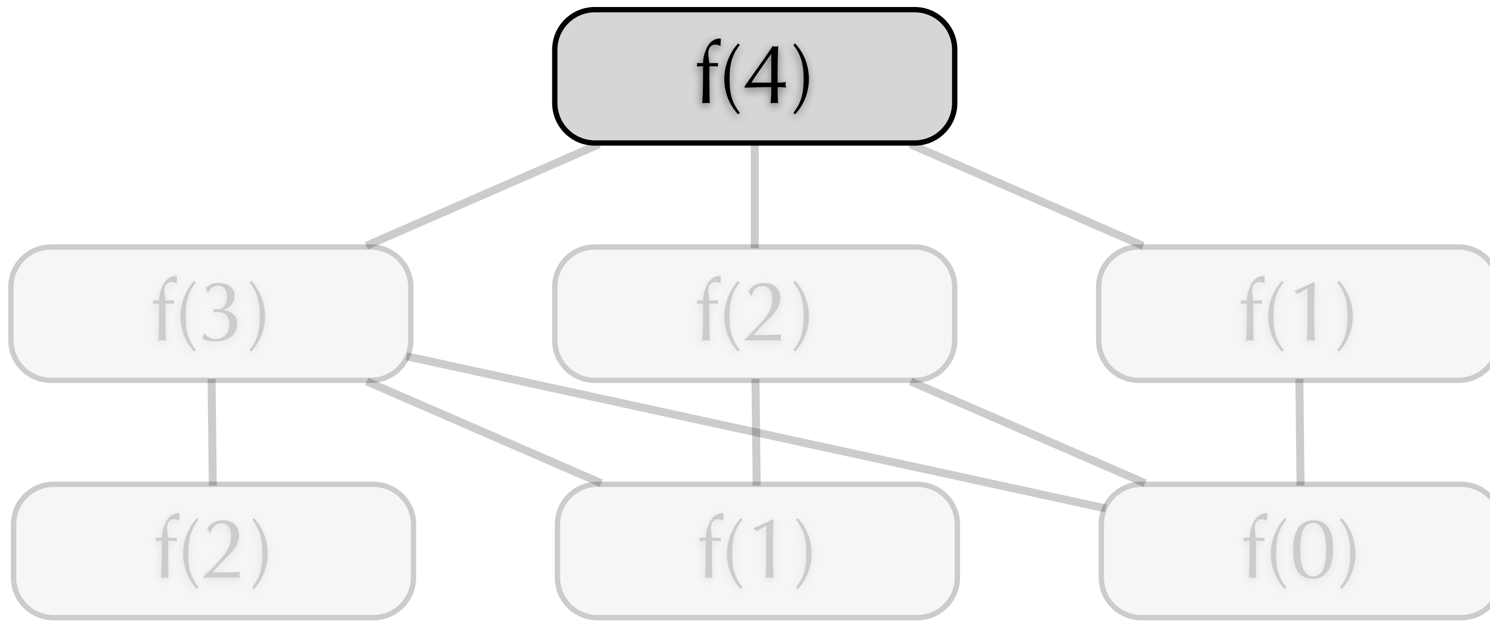
O(kn)?

Memoization

- Store the answers to the subproblems.
- All overlapping subproblems are only computed once.
  - Look up the table!
- Note: the authors of the textbook insist memoization is not a misspelling!

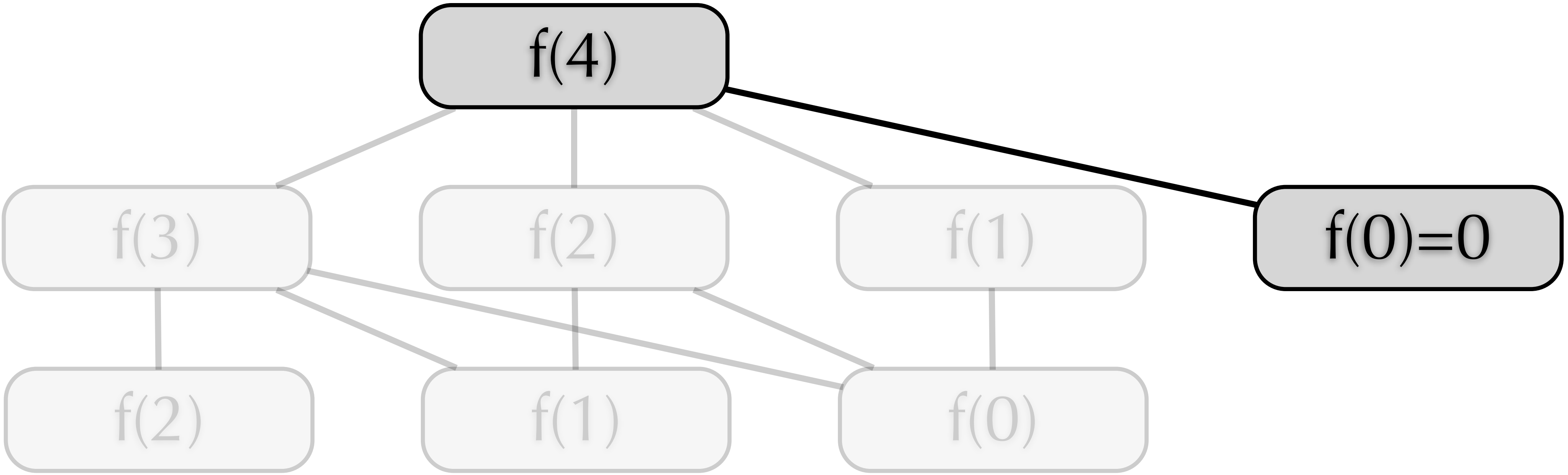
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)				



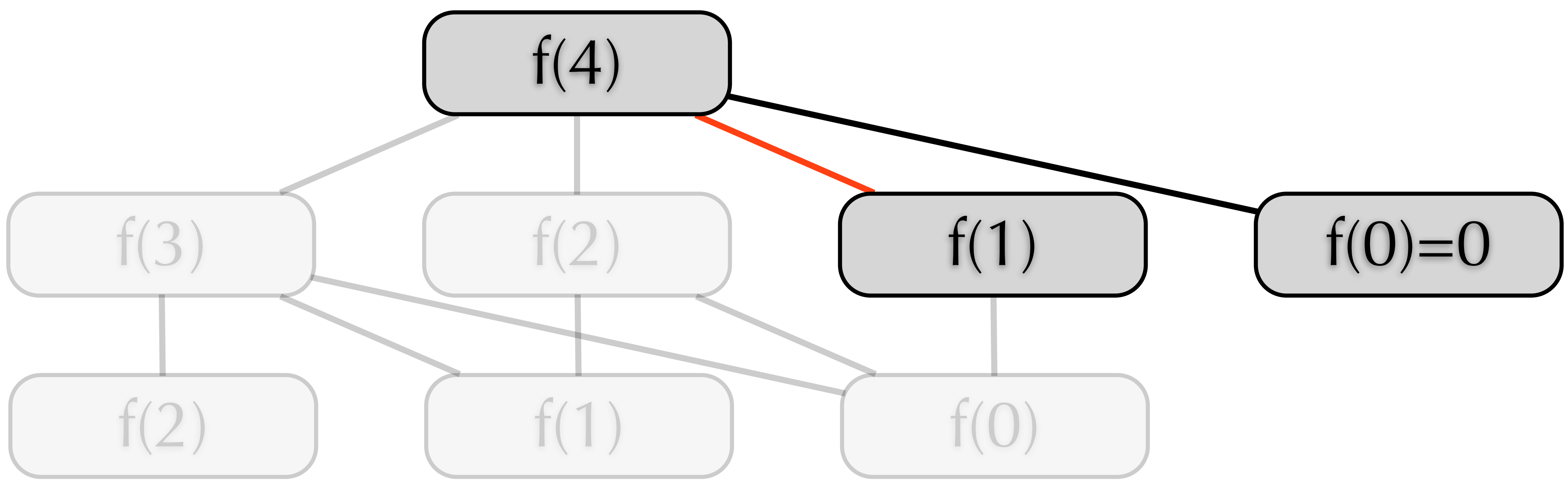
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)				



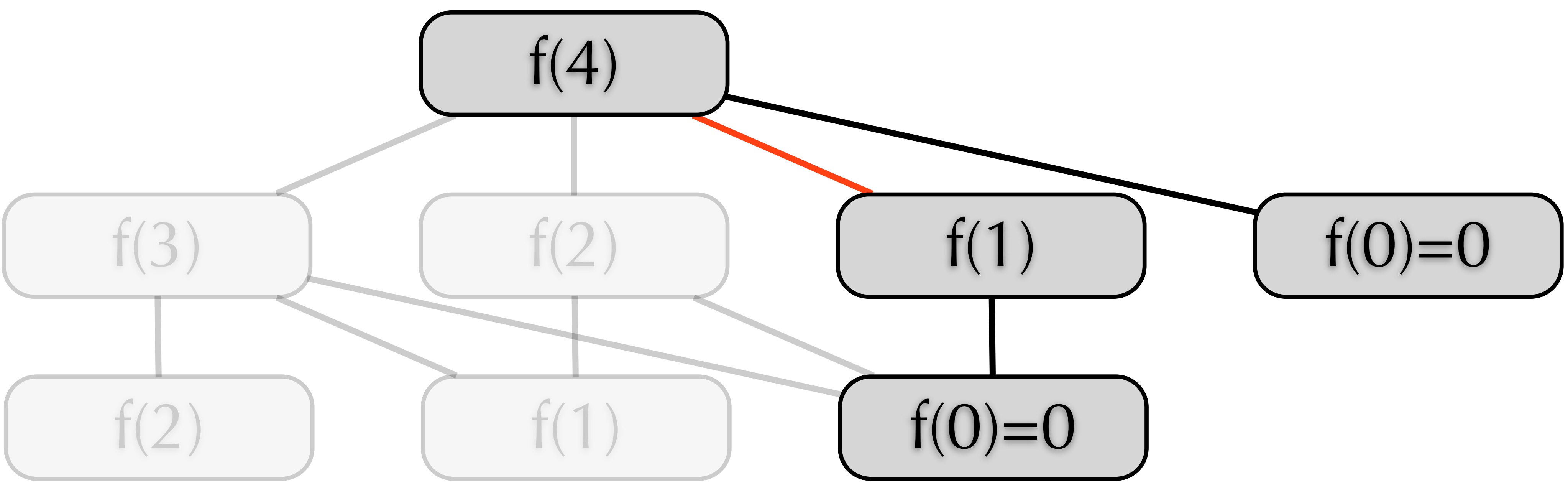
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)				



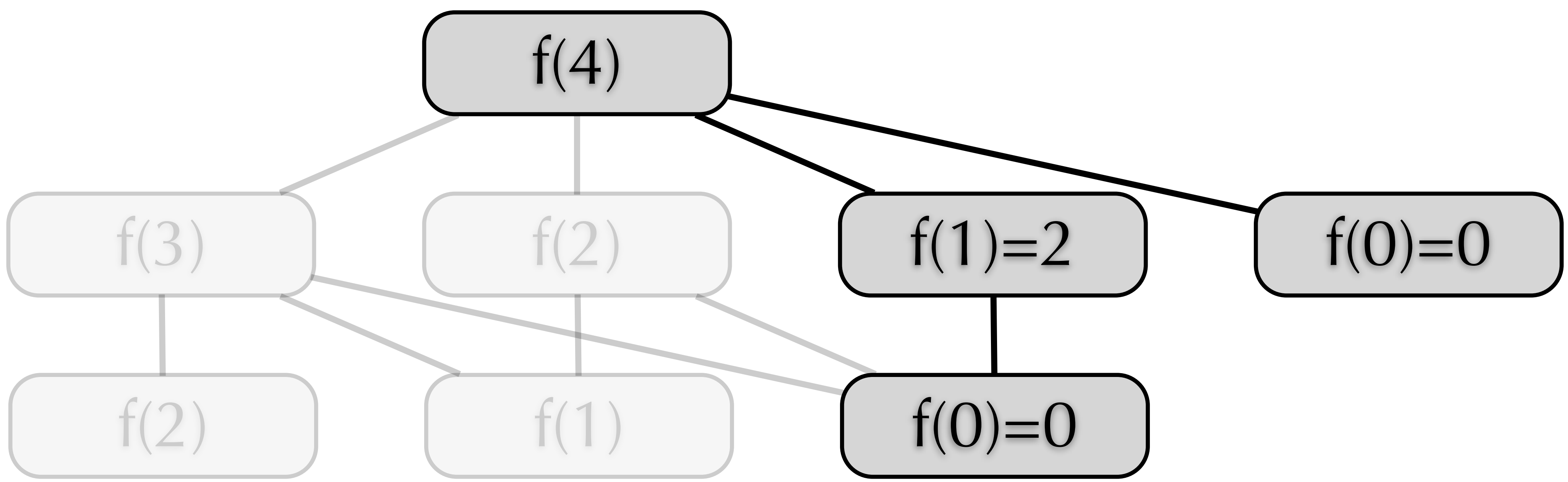
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)				



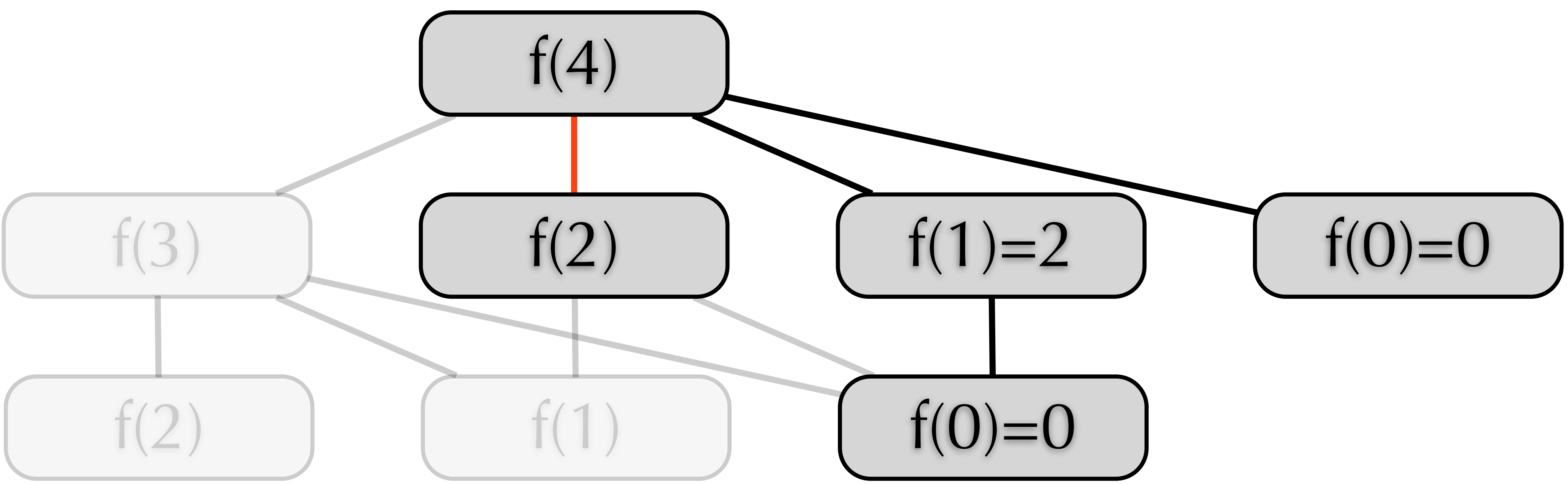
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2			



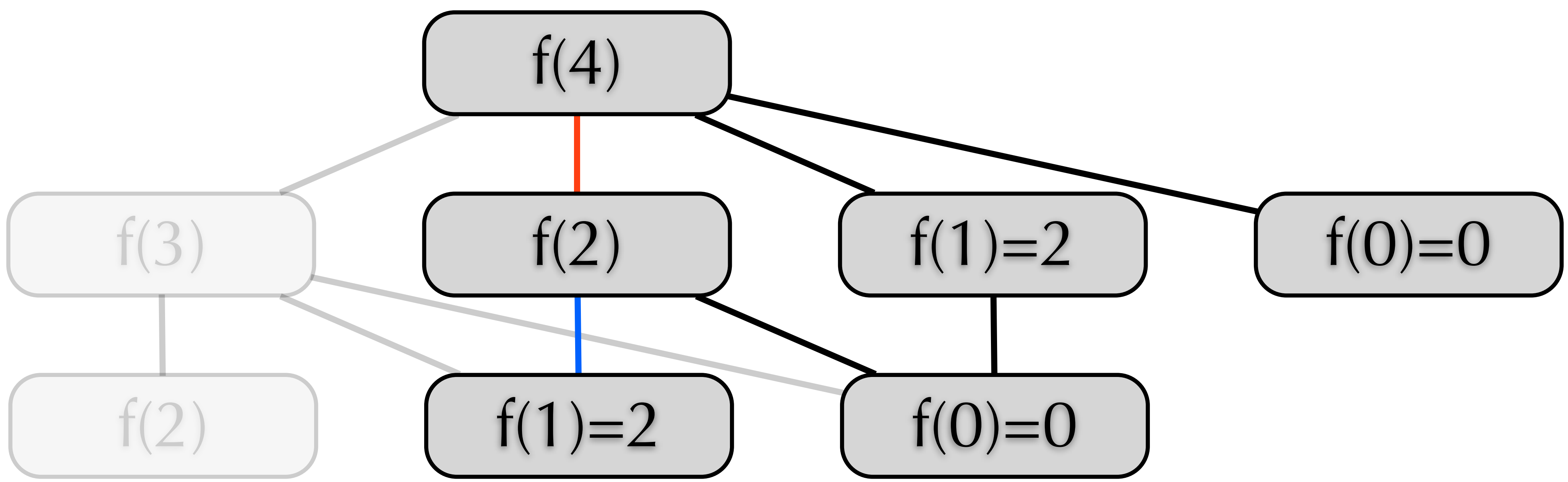
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2			



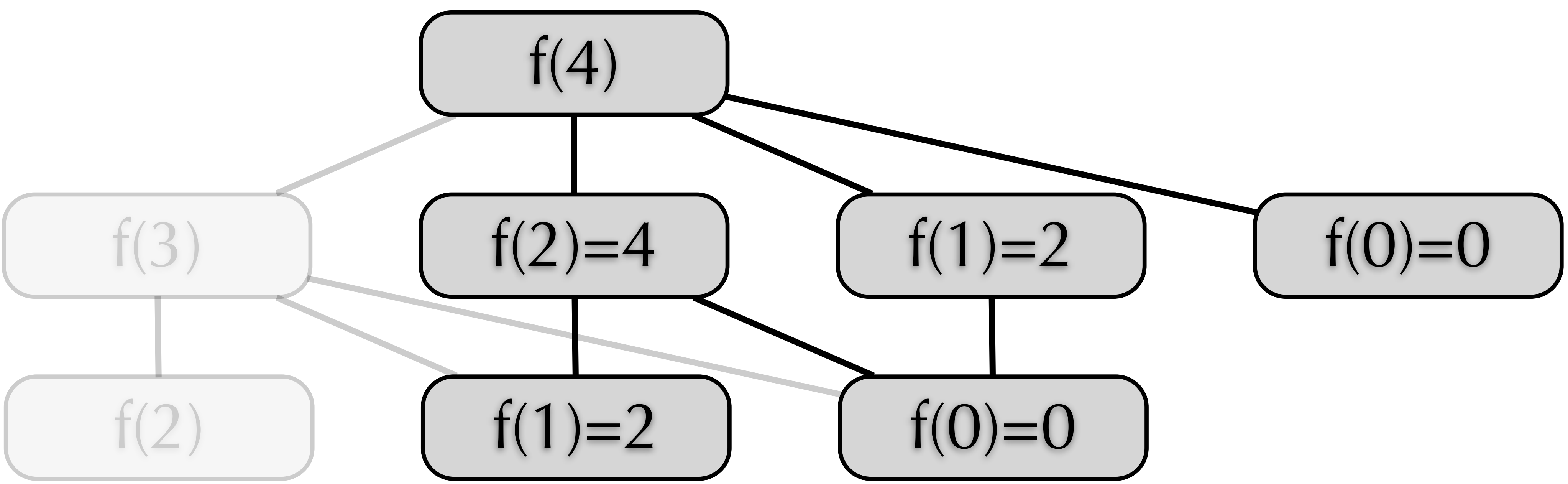
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2			



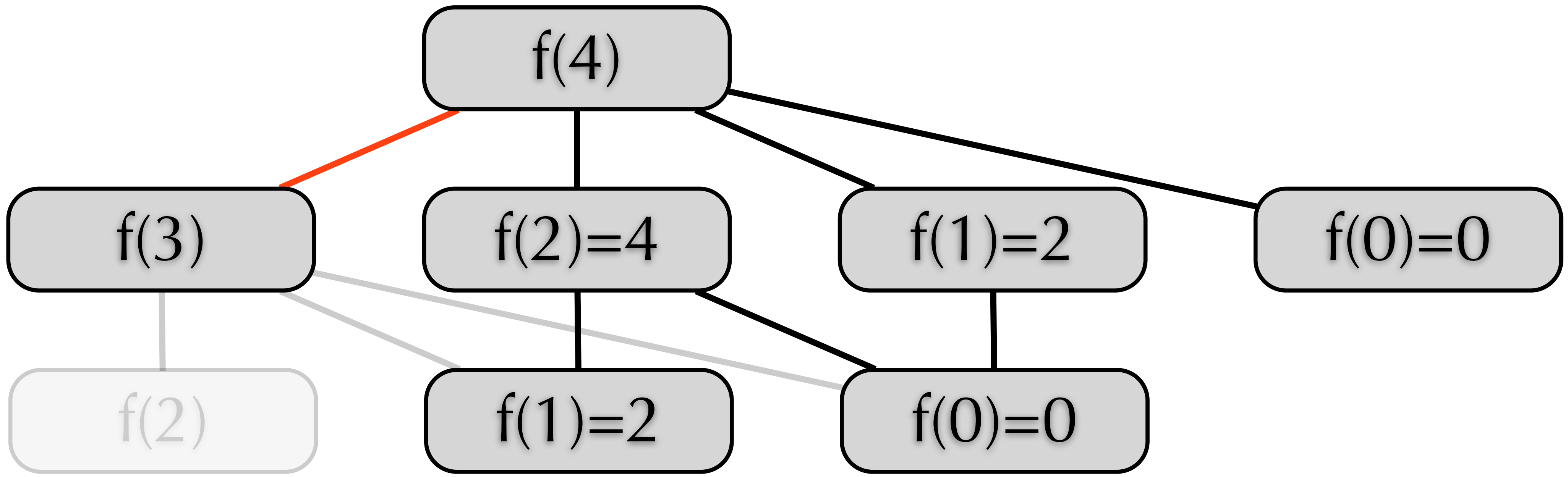
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4		



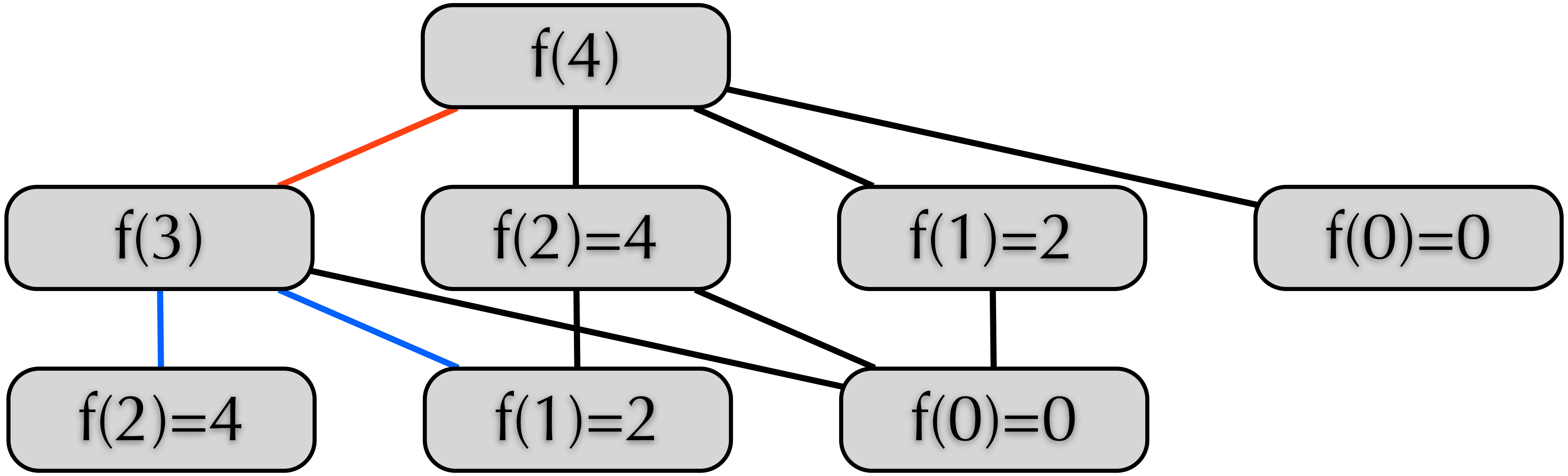
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4		



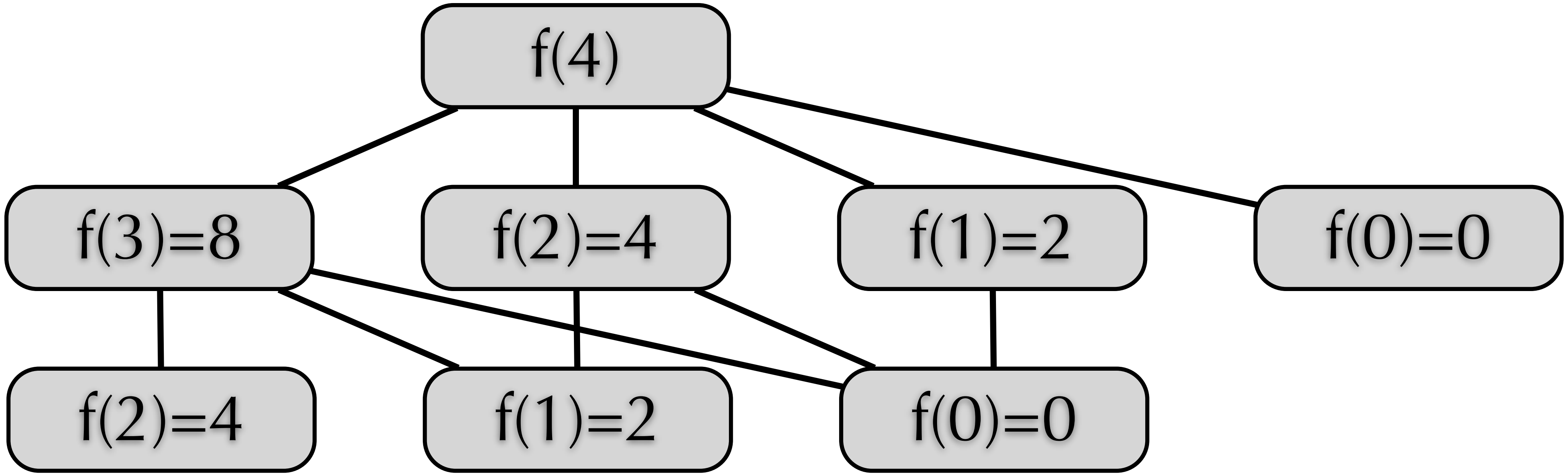
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4		



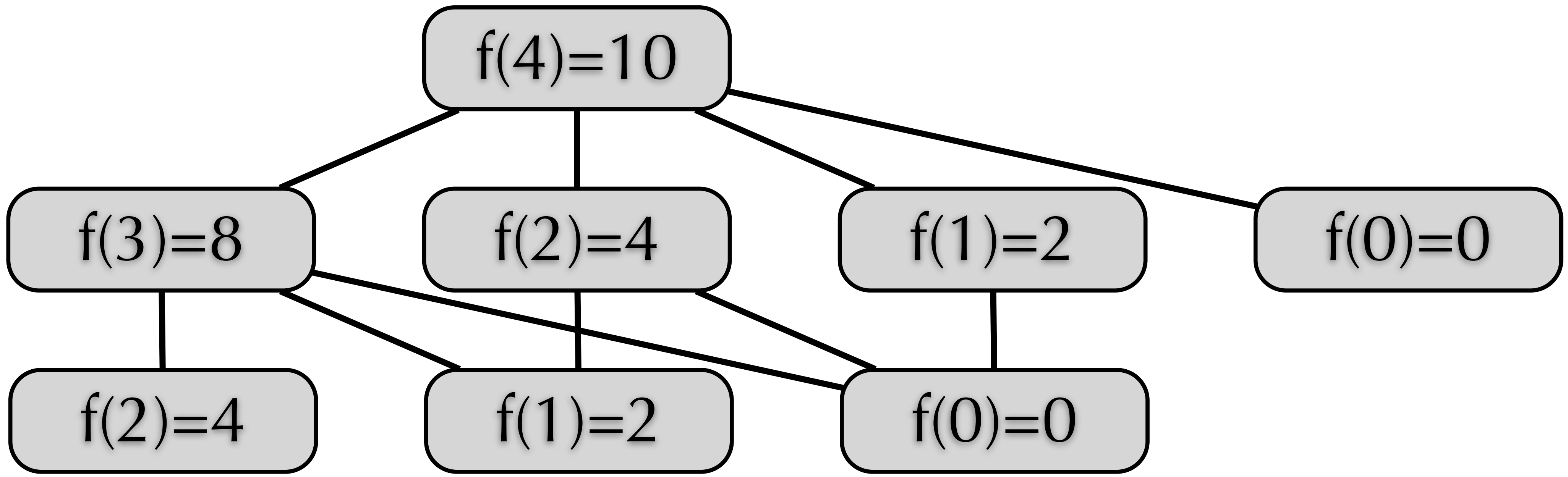
Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4	8	



Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4	8	10



Bottom-Up

- If the problem is not small enough, then we need to solve smaller subproblems.
- Solve the smallest subproblem first.
- Solve the other subproblems in ascending order.
- Iterative approach!
- Incremental approach??

Example: n=4, k=4

i	1	2	3	4
p_i	2	3	8	9
f(i)	2	4	8	10

$f(1)=\max_{1\leq i\leq 1}(f(1-i)+p_i)=2$

$f(2)=\max_{1\leq i\leq 2}(f(2-i)+p_i)=\max(2+2,0+3)=4$

$f(3)=\max_{1\leq i\leq 3}(f(3-i)+p_i)=\max(4+2,2+4,0+8)=8$

$f(4)=\max_{1\leq i\leq 4}(f(4-i)+p_i)=\max(8+2,4+4,2+8,0+9)=10$

Another Example
n=10, k=10

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	25
f(i)										

Practice: top-down & bottom up

Another Example
n=10, k=10

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	25
f(i)	1	5	8	10	13	17	18	22	25	27

The maximum revenue is found, but how to find the optimal cutting?

Divide and conquer: the LAST cut

Another Example
n=10, k=10

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	25
f(i)	1	5	8	10	13	17	18	22	25	27
d(i)	1	2	3	2	2	6	1	2	1	2

$$f(x) = \max_{1 \le i \le \min(k,x)} (f(x-i) + p_i) = f(x - d(i)) + p_{d(i)}$$

Note: d(i) isn't necessarily unique!

Another Example
n=10, k=10

i	1	2	3	4	5	6	7	8	9	10
p_i	1	5	8	9	10	17	17	20	24	25
f(i)	1	5	8	10	13	17	18	22	25	27
d(i)	1	2	3	2	2	6	1	2	1	2

d(10)=2, d(10–d(10))=d(8)=2, d(8–d(8))=d(6)=6

Optimal cutting for rods of length 10: (2,2,6)

Complexity

- #subproblems: $\Theta(n)$
- Constructing optimal solution: $O(n)$
- Total: $O(n^2)$

Dynamic Programming

- Characterize the structure of an optimal solution. A particular optimal solution is sufficient.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution.
- Construct an optimal solution from computed information.

Practice

- Suppose cutting is not free anymore!
- Q1. The cost is a constant. Ex: 1
- Q2. The cost is a function of the length of rod. (x is the length) Ex: $c(x)=x$
- Q3. The cost is a function of the length of rod and the cut point p. (Cut the rod into two pieces whose length are p and x–p) Ex: $\max(x-p,p)$

Matrix-Chain Multiplication

- $\langle A_1,\ldots,A_n \rangle$: a sequence of matrices
- $A_1A_2\ldots A_{n-1}A_n$ denotes the product.
- Matrix multiplication is associative.
  - ABC=(AB)C=A(BC)
- If A is a p-by-q matrix and B is a q-by-r matrix, then computing AB needs pqr multiplications.
  Ordinary matrix multiplication

Associativity

- There are 5 ways to compute $A_1A_2A_3A_4$.
  - $(A_1(A_2(A_3A_4)))$
  - $(A_1((A_2A_3)A_4))$
  - $((A_1A_2)(A_3A_4))$
  - $((A_1(A_2A_3))A_4)$
  - $(((A_1A_2)A_3)A_4)$

Matrix-Chain Multiplication

$$A_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, A_2 = \begin{pmatrix} d & e & f \end{pmatrix}, A_3 = \begin{pmatrix} g \\ h \\ i \end{pmatrix}, A_4 = \begin{pmatrix} j & k & \ell \end{pmatrix}$$

$$(A_1(A_2(A_3A_4))) : 3 \times 1 \times 3 + 1 \times 3 \times 3 + 3 \times 1 \times 3 = 27$$

$$(A_1((A_2A_3)A_4)) : 3 \times 1 \times 3 + 1 \times 3 \times 1 + 1 \times 1 \times 3 = 15$$

$$((A_1A_2)(A_3A_4)) : 3 \times 1 \times 3 + 3 \times 3 \times 3 + 3 \times 1 \times 3 = 45$$

$$((A_1(A_2A_3))A_4) : 3 \times 1 \times 1 + 1 \times 3 \times 1 + 3 \times 1 \times 3 = 15$$

$$(((A_1A_2)A_3)A_4) : 3 \times 1 \times 3 + 3 \times 3 \times 1 + 3 \times 1 \times 3 = 27$$

Matrix-Chain Multiplication

- Question: How to minimize the number of multiplications?
- Input: $(n,\langle p_0,p_1,\ldots,p_n\rangle)$
  - A_i is a p_{i-1}-by-p_i matrix.
- Output: The minimum number of multiplications & how to achieve

Idea

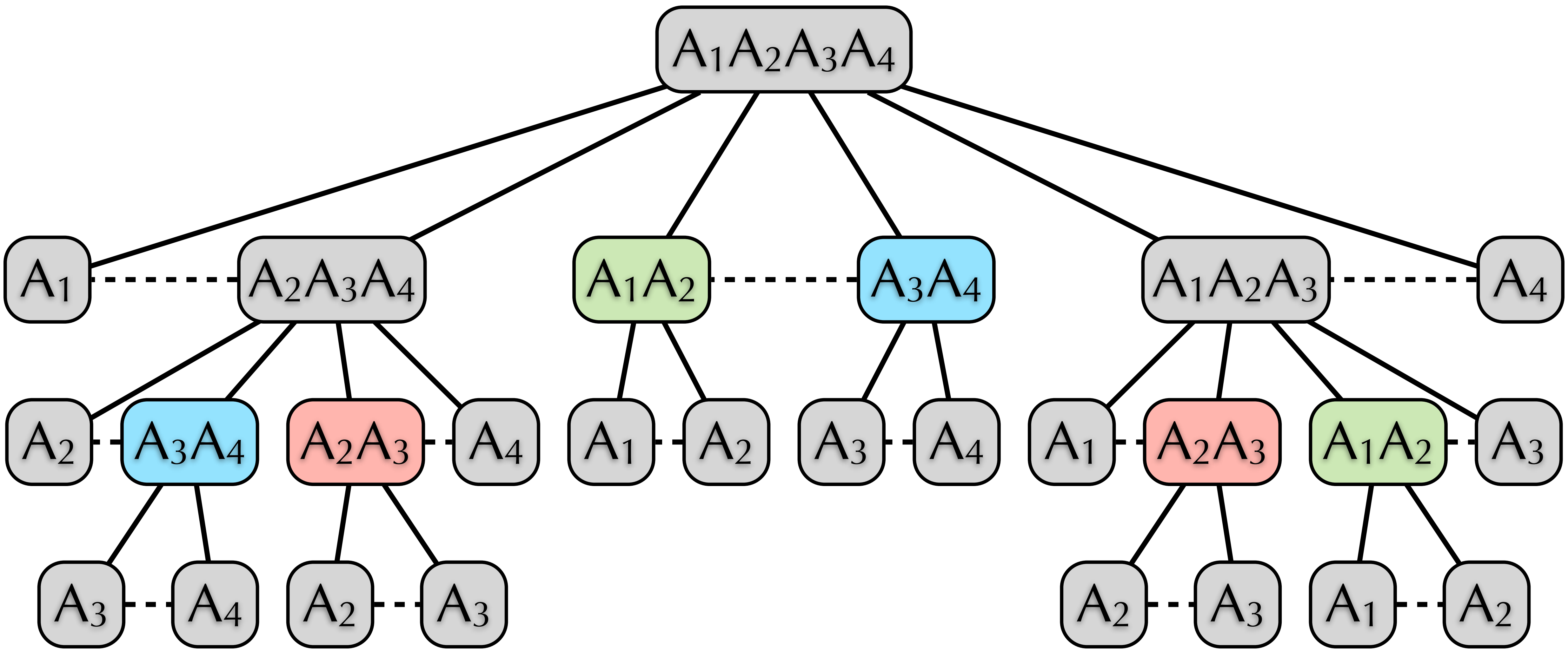
- The candidates of LAST multiplication
  - $A_1 \times (A_2 \ldots A_n)$, $(A_1 A_2) \times (A_3 \ldots A_n)$, …, $(A_1 \ldots A_{n-2}) \times (A_{n-1} A_n)$, $(A_1 \ldots A_{n-1}) \times A_n$
- If we know all optimal solutions of $A_1 \ldots A_i$ and $A_{i+1} \ldots A_n$ where $i \in \{1, \ldots, n-1\}$, then we can decide which multiplication should be the LAST one.

Divide and Conquer

- Termination: If n=1, return 0.
- Divide: $A_1 \ldots A_i$ & $A_{i+1} \ldots A_n$ for $i \in \{1, \ldots, n-1\}$
- Conquer: Compute the answers
  - Let f(i) be the answer of $A_1 \ldots A_i$
  - Let g(i) be the answer of $A_{i+1} \ldots A_n$
- Combine:
  return $\min_{1 \le i \le n-1}(f(i)+g(i)+p_0 p_i p_n)$

Example

Form of subproblems: $A_L \ldots A_R$



Optimal solution of $A_L \ldots A_R$: $h(L,R)$

Without Table

- C(n): The number of ways to multiply n matrices.
  - C(1)=C(2)=1
  - $C(n)=\Sigma_{1\leq i<n}C(i)C(n-i)$ for i>2
  - $C(n)=\Omega(n^{1.5}4^n)$

$$C(n) = \frac{1}{n}\binom{2n-2}{n-1}$$

- n-th Catalan number = C(n+1)
- How to solve the recurrence?
  - Generating function

Table

$$h(L,R)=\min_{L\leq i<R} h(L,i)+h(i+1,R)+p_{L-1}p_ip_R$$

L \ R	1	2	3	4
1	0			
2		0		
3			0	
4				0

$$\langle p_0,p_1,p_2,p_3,p_4\rangle=\langle 3,1,3,1,3\rangle$$

Table

$$h(L,R)=\min_{L\le i<R} h(L,i)+h(i+1,R)+p_{L-1}p_ip_R$$

L \ R	1	2	3	4
1	0	9		
2		0	3	
3			0	9
4				0

$$\langle p_0,p_1,p_2,p_3,p_4\rangle=\langle 3,1,3,1,3\rangle$$

Table

$$h(L,R)=\min_{L\leq i<R} h(L,i)+h(i+1,R)+p_{L-1}p_ip_R$$

L \ R	1	2	3	4
1	0	9	6	
2		0	3	
3			0	9
4				0

$0+3+3\times1\times1=6$

$9+0+3\times3\times1=18$

$\langle p_0,p_1,p_2,p_3,p_4\rangle=\langle 3,1,3,1,3\rangle$

Table

$$h(L,R)=\min_{L\le i<R} h(L,i)+h(i+1,R)+p_{L-1}p_ip_R$$

L \ R	1	2	3	4
1	0	9	6	
2		0	3	6
3			0	9
4				0

$0+9+1\times3\times3=18$

$3+0+1\times1\times3=6$

$$\langle p_0,p_1,p_2,p_3,p_4\rangle=\langle 3,1,3,1,3\rangle$$

Table

$$h(L,R)=\min_{L\leq i<R} h(L,i)+h(i+1,R)+p_{L-1}p_ip_R$$

L \ R	1	2	3	4
1	0	9	6	15
2		0	3	6
3			0	9
4				0

$0+6+3\times1\times3=15$

$9+9+3\times3\times3=45$

$0+6+3\times1\times3=15$

$\langle p_0,p_1,p_2,p_3,p_4\rangle=\langle 3,1,3,1,3\rangle$

Tables

h(L,R)

L \ R	1	2	3	4
1	0	9	6	15
2		0	3	6
3			0	9
4				0

d(L,R)

L \ R	1	2	3	4
1		1	1	1
2			2	3
3				3
4				

Optimal solution: $A_1((A_2A_3)A_4)$

Bottom-Up Order

L \ R	1	2	3	4
1	0	9	6	15
2		0	3	6
3			0	9
4				0

L \ R	1	2	3	4
1	0	9	6	15
2		0	3	6
3			0	9
4				0

Practice

h(L,R)

L \ R	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

d(L,R)

L \ R	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

$\langle p_0, p_1, p_2, p_3, p_4, p_5, p_6 \rangle = \langle 6, 7, 3, 1, 2, 4, 5 \rangle$

Practice

h(L,R)

L \ R	1	2	3	4	5	6
1	0	126	63	75	95	121
2		0	21	35	57	84
3			0	6	20	43
4				0	8	28
5					0	40
6						0

d(L,R)

L \ R	1	2	3	4	5	6
1		1	1	3	3	3
2			2	3	3	3
3				3	3	3
4					4	5
5						5
6						

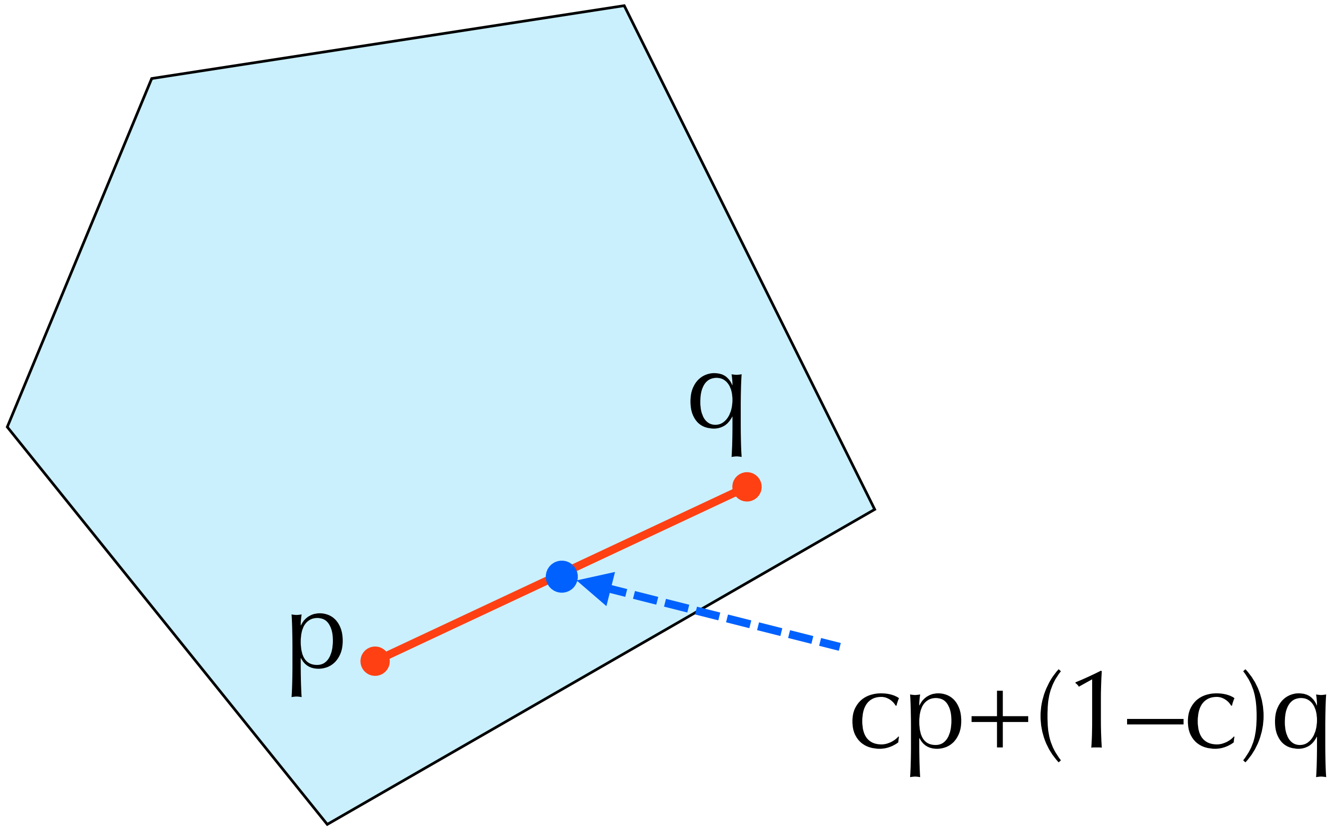
$\langle p_0,p_1,p_2,p_3,p_4,p_5,p_6\rangle=\langle 6,7,3,1,2,4,5\rangle$

Complexity

- #subproblems: $\Theta(n^2)$
- Compute the value of a subproblem: $O(n)$
- Construct optimal solution: $\Theta(n)$?
- Total: $O(n^3)$

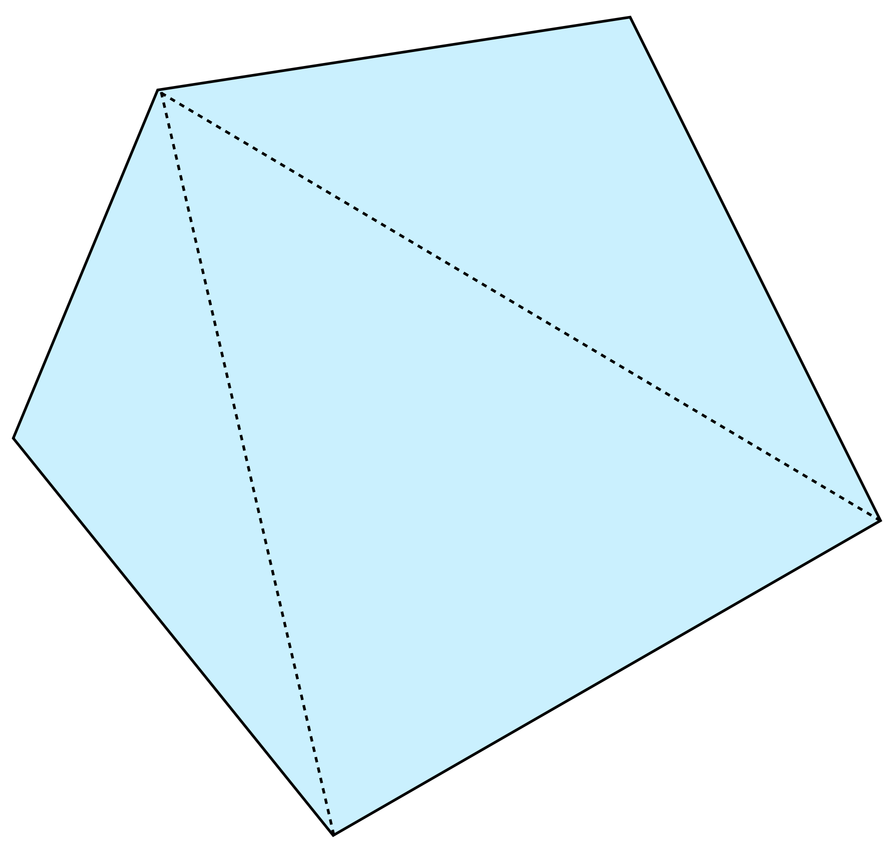
Convex Polygon Triangulation

- Convex polygon P: if $p=(x_p,y_p)$ and $q=(x_q,y_q)$ are two points inside P, then $cp+(1-c)q=(cx_p+(1-c)x_q,cy_p+(1-c)y_q)$ is inside P, for every $c\in[0,1]$.



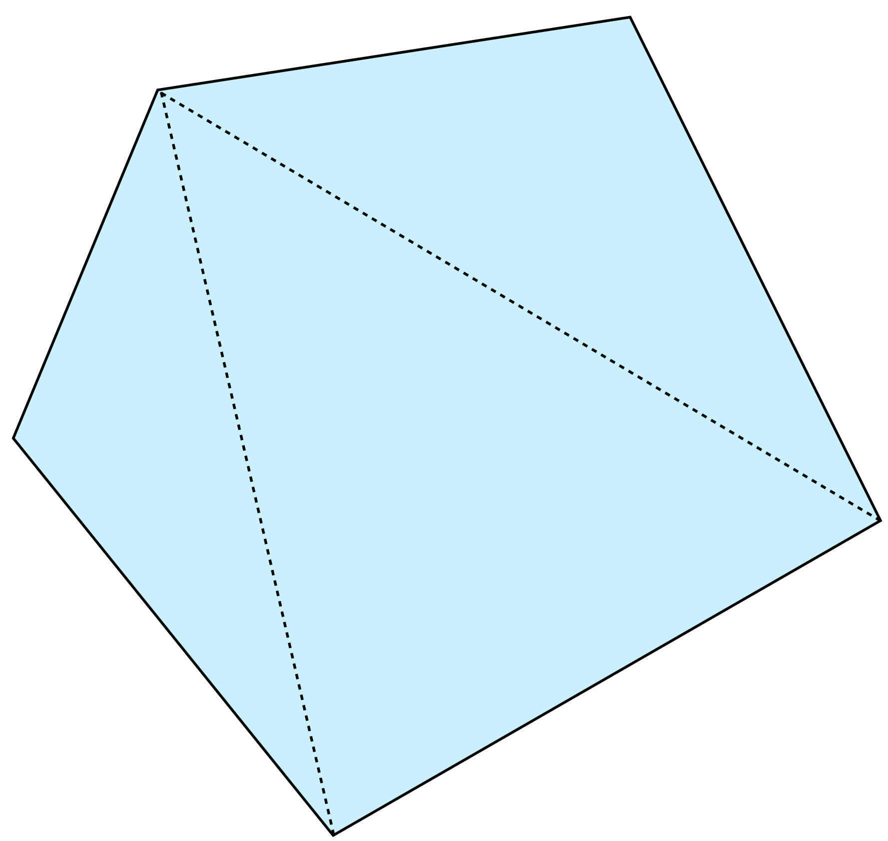
Convex Polygon Triangulation

- Triangulation: Partition the polygon into triangles.
- Cost Ex: length of the line segments, the variance of the area of the triangles, etc.



Convex Polygon Triangulation

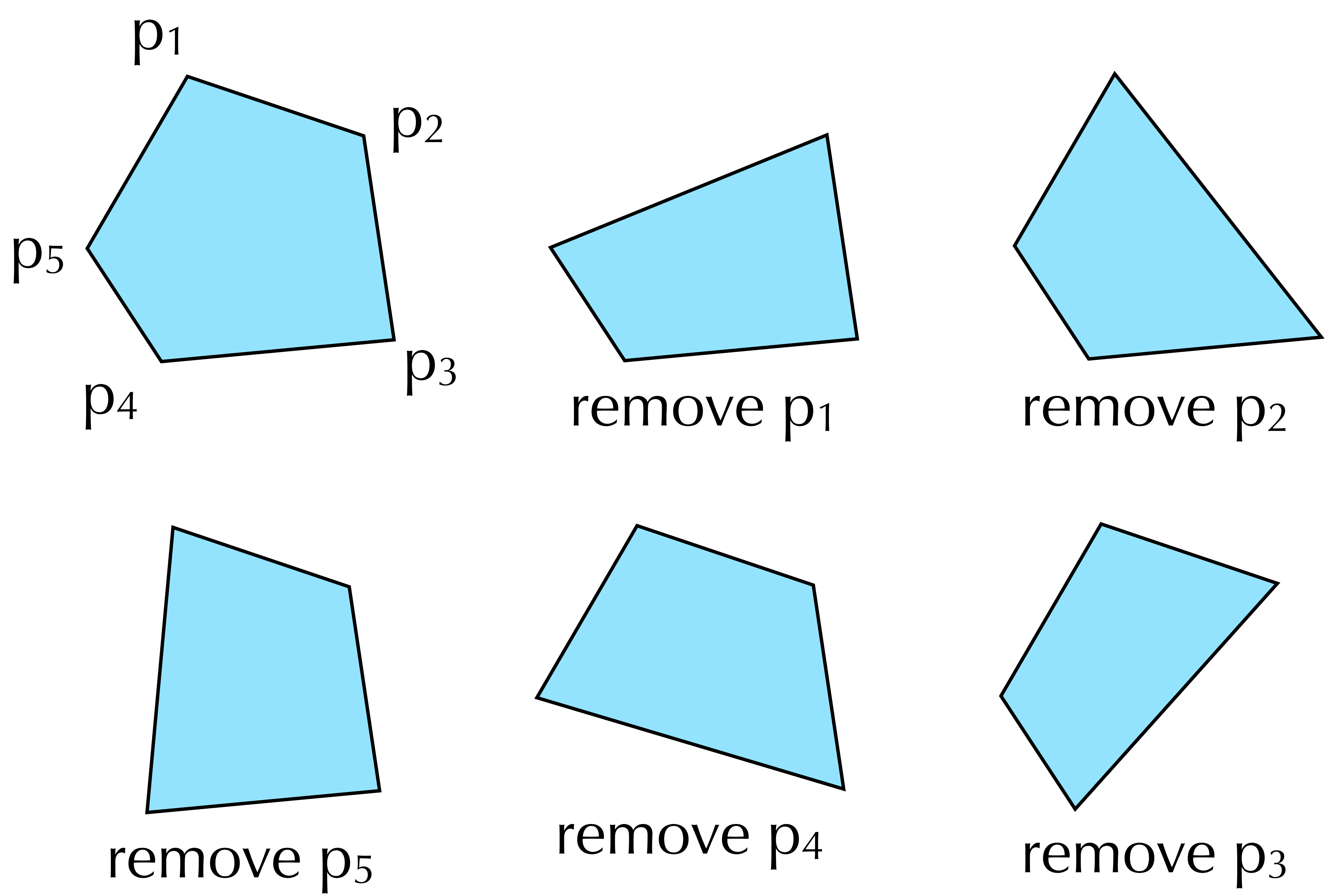
- Goal: Find an optimal partition achieve minimum/maximum cost
- How?



Convex Polygon Triangulation

- Homework 2-2
  - Input: $(n, p_1, \ldots, p_n)$ where $p_i=(x_i,y_i)$ are given in the clockwise order.
- Cost: length of the line segments
- Goal: minimization
- How to solve?
  - The idea of "the last cut"

Last Cut



p1
p2
p5
p3
p4
remove p1
remove p2
remove p5
remove p4
remove p3

Convex Polygon Triangulation

- Termination: If $n=3$, return 0.
- Divide-and-Conquer: Solve subproblems
  - $s_1=(n-1, p_2, \ldots, p_n)$
  - $s_i=(n-1, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$ $i \in (1,n)$
  - $s_n=(n-1, p_1, \ldots, p_{n-1})$
- Combine: return $\min_{1 \leq i \leq n}(\mathrm{opt}(s_i)+L_i)$ where L_i is the length of line segment connecting p_{i-1} and p_{i+1}.

Convex Polygon Triangulation

- How many subproblems?
  - All ≥3-subsets of $\{p_1,\ldots,p_n\}$ $\leq 2^n$
- Solving a subproblem needs to choose the best answer from n candidates.
- Time complexity: $O(n2^n)$
- Capable to solve the basic case
- How to solve the hard case?
  - Hint: matrix-chain multiplication

Elements of DP

- Optimal substructure
  - An optimal solution contains within it optimal solutions of subproblems
- Overlapping subproblems
  - What if the subproblems do not overlap?

Longest Common Subsequence

- For sequence $\langle a_1,\ldots,a_n \rangle$:
  - $\langle c_1,\ldots,c_k \rangle$ is a subsequence of $\langle a_1,\ldots,a_n \rangle$ if $c_j = a_{i[j]}$ where $1 \le i[1] < i[2] < \ldots < i[k] \le n$.
- $\langle c_1,\ldots,c_k \rangle$ is a common subsequence of $\langle a_1,\ldots,a_n \rangle$ and $\langle b_1,\ldots,b_m \rangle$ if
  - $\langle c_1,\ldots,c_k \rangle$ is a subsequence of $\langle a_1,\ldots,a_n \rangle$.
  - $\langle c_1,\ldots,c_k \rangle$ is a subsequence of $\langle b_1,\ldots,b_m \rangle$.

Compute the Length

- Input: $s=(n,m,a_1,\ldots,a_n,b_1,\ldots,b_m)$
- Termination: If $n=0$ or $m=0$, return 0.
- Divide-and-Conquer: Solve subproblems
  - $s_A=(n-1,m,a_1,\ldots,a_{n-1},b_1,\ldots,b_m)$
  - $s_B=(n,m-1,a_1,\ldots,a_n,b_1,\ldots,b_{m-1})$
  - $s_E=(n-1,m-1,a_1,\ldots,a_{n-1},b_1,\ldots,b_{m-1})$
- Combine: If $a_n=b_m$ then return $\max(opt(s_A),opt(s_B),opt(s_E)+1)$. Otherwise, return $\max(opt(s_A),opt(s_B))$.

Subproblems

- $s_A=(n-1,m,a_1,\ldots,a_{n-1},b_1,\ldots,b_m)$:
  - $opt(s)=opt(s_A)$ if a_n is not in LCS.
- $s_B=(n,m-1,a_1,\ldots,a_n,b_1,\ldots,b_{m-1})$
  - $opt(s)=opt(s_B)$ if b_m is not in LCS.
- $s_E=(n-1,m-1,a_1,\ldots,a_{n-1},b_1,\ldots,b_{m-1})$
  - $opt(s)=opt(s_E)+1$ if $a_n=b_m$ is in LCS.
- Homework: Show that if $a_n=b_m$, then $opt(s_E)+1\geq\max(opt(s_A),opt(s_B))$.

Example

	i	0	1	2	3	4	5	6	7
j		a_i	A	B	C	B	D	A	B
0	b_j								
1	B								
2	D								
3	C								
4	A								
5	B								
6	A								

Terminal Condition

	i	0	1	2	3	4	5	6	7
j		a_i	A	B	C	B	D	A	B
0	b_j	0	0	0	0	0	0	0	0
1	B	0							
2	D	0							
3	C	0							
4	A	0							
5	B	0							
6	A	0							

Compute the Values

	i	0	1	2	3	4	5	6	7
j		a_i	A	B	C	B	D	A	B
0	b_j	0	0	0	0	0	0	0	0
1	B	0	←0	↖1	←1	↖1	←1	←1	↖1
2	D	0	←0	↑1	←1	←1	↖2	←2	←2
3	C	0	←0	↑1	↖2	←2	←2	←2	←2
4	A	0	↖1	←1	↑2	←2	←2	↖3	←3
5	B	0	↑1	↖2	←2	↖3	←3	←3	↖4
6	A	0	↖1	↑2	←2	↑3	←3	↖4	←4

Construct Solution

LCS of ABCBDAB and BDCABA: BCBA

	i	0	1	2	3	4	5	6	7
j		a_i	A	B	C	B	D	A	B
0	b_j	0	0	0	0	0	0	0	0
1	B	0	←0	↖1	←1	↖1	←1	←1	↖1
2	D	0	←0	↑1	←1	←1	↖2	←2	←2
3	C	0	←0	↑1	↖2	←2	←2	←2	←2
4	A	0	↖1	←1	↑2	←2	←2	↖3	←3
5	B	0	↑1	↖2	←2	↖3	←3	←3	↖4
6	A	0	↖1	↑2	←2	↑3	←3	↖4	←4

Time Complexity

- #subproblems: $\Theta(nm)$
- Compute the value of a subproblem: $\Theta(1)$
- Construct optimal solution: $\Theta(n+m)$
- Total: $\Theta(nm)$

Longest Non-Decreasing Subsequence

- HW2-1
- Given an array A[1..n]
- Output the length of the longest non-decreasing subsequence of A:
  - Non-decreasing subsequence: $A[i_1] \leq A[i_2] \leq \ldots \leq A[i_{k-1}] \leq A[i_k]$
- How to compute the sequence?
  - Sorting + LCS: $O(n^2)$

Solution by DP

- Let L[k] be the maximum length of non-decreasing subsequences of A[1..k] which ends at A[k].
- L[1]=1
- $L[k]=\max(1, \max_{j<k, A[j]\leq A[k]}(L[j]+1))$
- The answer: $\max_{1\leq k\leq n}(L[k])$
- Time complexity? $O(n^2)$
- Space complexity: $\Theta(n)$

Examples

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	2	0	5	6	4	9	8	1	7
L[i]										
d[i]										

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	9	0	8	6	7	3	9	0	4
L[i]										
d[i]										

Compute the Values

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	2	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
d[i]	0	1	0	2	4	2	5	5	1/3	5

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	9	0	8	6	7	3	9	0	4
L[i]	1	2	1	2	2	3	2	4	2	3
d[i]	0	1	0	1	1	5	1/3	6	3	7/9

Construct Solution

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	2	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
d[i]	0	1	0	2	4	2	5	5	1/3	5

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	9	0	8	6	7	3	9	0	4
L[i]	1	2	1	2	2	3	2	4	2	3
d[i]	0	1	0	1	1	5	1/3	6	3	7/9

Construct Solution

LNDS of ⟨2,2,0,5,6,4,9,8,1,7⟩: ⟨2,2,5,6,7⟩

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	2	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
d[i]	0	1	0	2	4	2	5	5	1/3	5

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	9	0	8	6	7	3	9	0	4
L[i]	1	2	1	2	2	3	2	4	2	3
d[i]	0	1	0	1	1	5	1/3	6	3	7/9

LNDS of ⟨2,9,0,8,6,7,3,9,0,4⟩: ⟨2,6,7,9⟩

Complexity

- #subproblems: $\Theta(n)$
- Compute the value of a subproblem: $\Theta(n)$
- Construct optimal solution: $\Theta(n)$
- Total: $\Theta(n^2)$

0-1 Knapsack Problem

- A bad guy robs a store and finds n items
  - The i-th item has weight w_i kg.
  - The i-th item has value v_i dollars.
  - Each item cannot be divided into smaller pieces.
  - He can only carry at most W with his knapsack.
- How to achieve the maximum total value?

0-1 Knapsack Problem

- Input: $s=(n,W,v_1,w_1,\ldots,v_n,w_n)$
- Termination: If $n=0$ or $W\leq 0$, return 0.
- Divide-and-Conquer: Solve subproblems
  - $s_C=(n-1,W-w_n,v_1,w_1,\ldots,v_{n-1},w_{n-1})$
  - $s_P=(n-1,W,v_1,w_1,\ldots,v_{n-1},w_{n-1})$
- Combine: $\max(\mathrm{opt}(s_C)+v_n,\mathrm{opt}(s_P))$.

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6									

Terminal Condition

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P							
2	0								
3	0								
4	0								
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P						
2	0								
3	0								
4	0								
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P	5,C					
2	0								
3	0								
4	0								
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P	5,C	5,C	5,C	5,C	5,C	5,C
2	0	0,P	0,P	5,P	6,C	6,C	6,C	11,C	
3	0								
4	0								
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P	5,C	5,C	5,C	5,C	5,C	5,C
2	0	0,P	0,P	5,P	6,C	6,C	6,C	11,C	11,C
3	0	0,P	0,P	5,P	6,P	6,P	6,P	11,P	11,P
4	0	0,P	0,P	5,P	6,P	6,C|P			
5	0								
6	0								

Compute the Values

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P	5,C	5,C	5,C	5,C	5,C	5,C
2	0	0,P	0,P	5,P	6,C	6,C	6,C	11,C	11,C
3	0	0,P	0,P	5,P	6,P	6,P	6,P	11,P	11,P
4	0	0,P	0,P	5,P	6,P	6,C|P	6,C|P	11,P	11,C|P
5	0	0,P	3,C	5,P	6,P	8,C	9,C	11,P	11,P
6	0	0,P	3,C|P	5,P	6,C|P	8,C|P	9,C|P	11,C|P	12,C

Construct Solution

(6,8,5,3,6,4,4,3,6,5,3,2,3,2)

i\W	≤0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0,P	0,P	5,C	5,C	5,C	5,C	5,C	5,C
2	0	0,P	0,P	5,P	6,C	6,C	6,C	11,C	11,C
3	0	0,P	0,P	5,P	6,P	6,P	6,P	11,P	11,P
4	0	0,P	0,P	5,P	6,P	6,C|P	6,C|P	11,P	11,C|P
5	0	0,P	3,C	5,P	6,P	8,C	9,C	11,P	11,P
6	0	0,P	3,C|P	5,P	6,C|P	8,C|P	9,C|P	11,C|P	12,C

Take items 2,5,6

Time Complexity

- #subproblems: $\Theta(nW)$
- Compute the value of a subproblem: $\Theta(1)$
- Construct optimal solution: $\Theta(n)$
- Total: $\Theta(nW)$
- Problems:
  - What if $W>>n$? What if $W>2^n$?
  - What if W is not integral?