Dynamic Programming

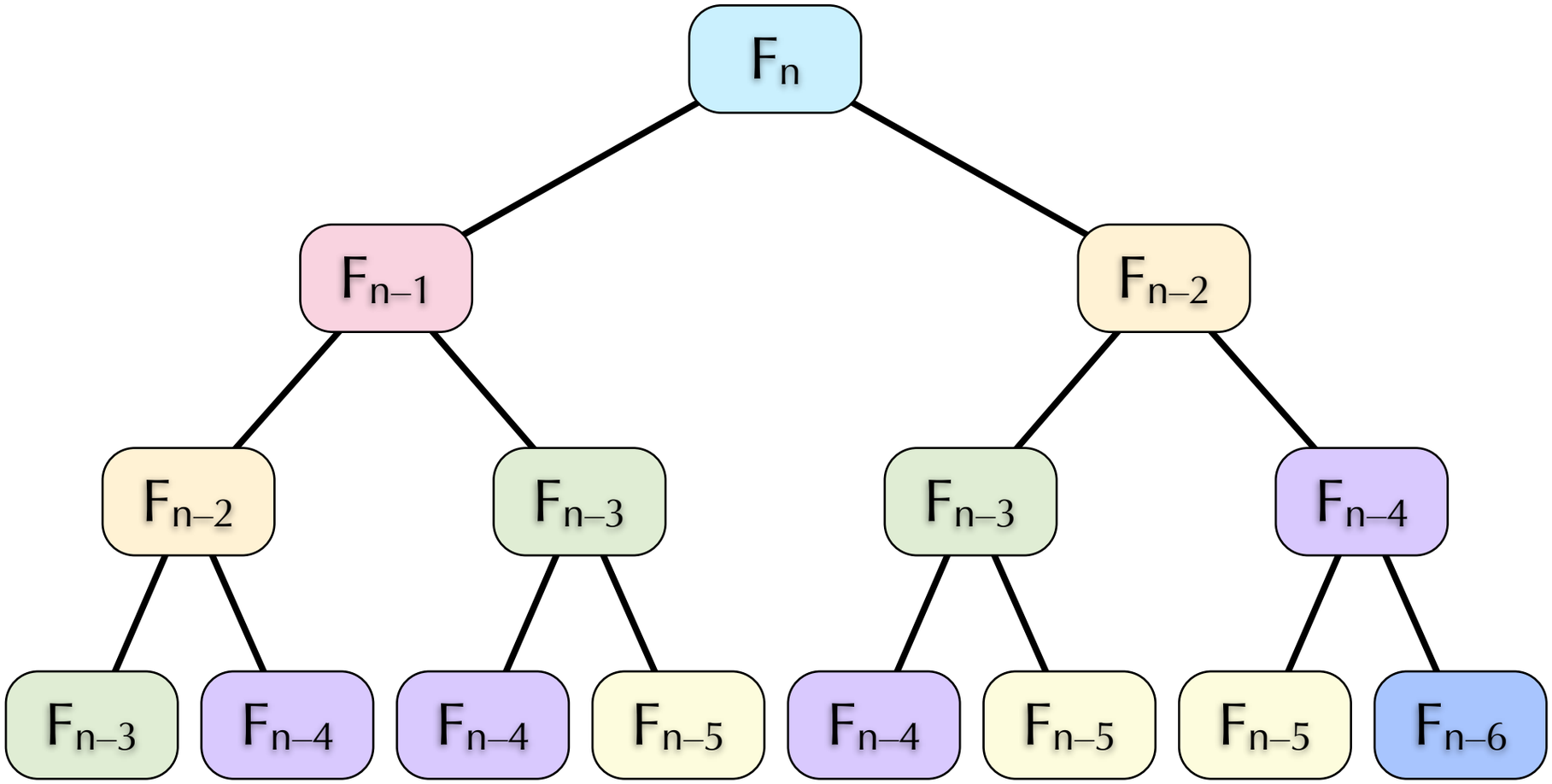
Spirit

- Never compute anything twice!
- Example:
  - Fibonacci numbers
  - Prefix sum
  - Longest increasing subsequence
  - Traveling salesperson problem
  - Weighted perfect matching
  - Subset sum

Divide & Conquer?

- Fibonacci numbers:
  $F_1=F_2=1$, $F_n=F_{n-1}+F_{n-2}$ for $n\geq 3$.
- D&C: $T(n)=T(n-1)+T(n-2)+O(1)=O(F_n)$
  - $F_n=\Omega(1.6^n)$
- Task: Write a program to compute the last 9 digits of F_n for $1\leq n\leq 10^6$.
  - D&C = TLE?!

Overlapping Subproblems



Dynamic Programming

- Solve each overlapping subproblem only once.
- How?
  - Recursive implementation by memoization (Easier)
  - Iterative implementation (Harder)

Memoization

```
int F9(int n)
{
   static int memo[1000001]={0,1,1};
   if(memo[n]==0)
      memo[n]=(F9(n-1)+F9(n-2))%1000000000;
   return memo[n];
}
```

Iterative Implementation

```
int F9(int n)
{
  int i, a=1, b=1, c=1;
  for(i=3;i<=n;i++)
    c=a+b, a=b, b=c%1000000000;
  return c;
}
```

Simpler Solution?

- Harder task: Write a program to compute the last 9 digits of F_n for $1 \le n \le 10^{666}$.
- Observation:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

- So,

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

1D Range Sum

- There is an sequence $s_1,\ldots,s_n$.
- Problem: Given two integers i and j such that $1 \le i \le j \le n$, compute $s_i+\ldots+s_j$.
- If we only need to compute once, then just do it as asked. O(j–i) additions
- But what if we need to compute this kind of question many many times?

Range Sum

1	2	3	4	5	6	7	8	9
0	1	1	1	0	2	1	1	0

1	2	3	4	5	6	7	8	9	
0	1	2	3	3	5	6	7	7	1
	1	2	3	3	5	6	7	7	2
		1	2	2	4	5	6	6	3
			1	1	3	4	5	5	4
				0	2	3	4	4	5
					2	3	4	4	6
						1	2	2	7
							1	1	8
								0	9

Straightforward: Store all range sums in a 2D table.

Space: $O(n^2)$

Query: $O(1)$

Prefix Sums

- Let $p_0=0$ and $p_i=s_1+s_2+\ldots+s_i$. We say $(p_0,\ldots,p_n)$ is the prefix sums of $(s_1,\ldots,s_n)$.
  - Note: $p_i=p_{i-1}+s_i$. This fact allow us to evaluate $(p_0,\ldots,p_n)$ in $O(n)$.
- Compute $s_i+\ldots+s_j$: use p_j-p_{i-1}.
- Build time: $O(n)$
- Query: $O(1)$
- Efficiently update?

Prefix Sum: Update s_5=1

	0	1	2	3	4	5	6	7	8	9
A[i]		0	1	1	1	0	2	1	1	0
P[i]	0	0	1	2	3	3	5	6	7	7

	0	1	2	3	4	5	6	7	8	9
A[i]		0	1	1	1	1	2	1	1	0
P[i]	0	0	1	2	3	4	6	7	8	8

Prefix Sums: 2D

- Consider an n-by-n matrix $A=(a_{i,j})$.
- Let $p_{0,0}=0$ and $p_{i,j}=\Sigma_{0<x\le i}\Sigma_{0<y\le j}a_{i,j}$ for $i>0$ or $j>0$. We say $P=(p_{i,j})$ is the prefix sums of A.
  - Note: $p_{i,j}=p_{i-1,j}+p_{i,j-1}+a_{i,j}-p_{i-1,j-1}$. This fact allow us to evaluate P in $O(n^2)$.
- How to compute $\Sigma_{L\le x\le R}\Sigma_{B\le y\le U}a_{i,j}$?
  - $(p_{R,U})-(p_{L-1,U})-(p_{R,B-1})+(p_{L-1,B-1})$

Prefix Sums: 2D

A	1	2	3	4	5
1	0	0	0	0	1
2	0	1	1	0	0
3	0	0	0	2	0
4	1	0	1	0	0
5	0	1	0	0	1

P	1	2	3	4	5
1					
2					
3					
4					
5					

Prefix Sums: 2D

A	1	2	3	4	5
1	0	0	0	0	1
2	0	1	1	0	0
3	0	0	0	2	0
4	1	0	1	0	0
5	0	1	0	0	1

P	1	2	3	4	5
1	0	0	0	0	1
2	0	1	**2**	**2**	3
3	0	1	**2**	**4**	5
4	1	2	4	6	7
5	1	3	5	7	9

Prefix Sums: 2D

A	1	2	3	4	5
1	0	0	0	0	1
2	0	1	1	0	0
3	0	0	0	2	0
4	1	0	1	0	0
5	0	1	0	0	1

P	1	2	3	4	5
1	0	0	0	0	1
2	0	1	2	2	3
3	0	1	2	4	5
4	1	2	4	6	7
5	1	3	5	7	9

Prefix Sums: 2D

A	1	2	3	4	5
1	0	0	0	0	1
2	0	1	1	0	0
3	0	0	0	2	0
4	1	0	1	0	0
5	0	1	0	0	1

P	1	2	3	4	5
1	0	0	0	0	1
2	0	1	2	2	3
3	0	1	2	4	5
4	1	2	4	6	7
5	1	3	5	7	9

Longest Increasing Subsequence

- Given an array A[1..n]
- Output the length of the longest increasing subsequence of A:
  - Increasing subsequence: $A[i_1]<A[i_2]<\ldots<A[i_{k-1}]<A[i_k]$ where $i_1<i_2<\ldots<i_{k-1}<i_k$
- How to compute the LIS?

Solution by DP

$\max_{j \in \varnothing} L[j] = -\infty$

- Let L[k] be the maximum length of increasing subsequences of A[1..k] which ends at A[k].
- L[1]=1
- L[k]=max($\max_{j<k, A[j]<A[k]}(L[j]+1)$, 1)
- The answer: $\max_{1 \le k \le n}(L[k])$
- Time complexity? $O(n^2)$
- How to build the LIS?

Example

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

Example

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

Example

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

Example

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

More Efficient Method

$\max_{j\in\varnothing} L[j]=-\infty$

- How to improve the algorithm to O(nlogn)?
  - $\max_{j<k,A[j]<A[k]}(L[j]+1)$ can be done in O(logn) time
  - By prune and search!
  - Let S[k,p]={x: x<k & L[x]=p}. An increasing subsequence of length p+1 ends at A[k] if $A[k]>\min_{x\in S[k,p]}A[x]$.

Example 1

$\min_{x \in S[10,1]} A[x] = 0$

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	3	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
∈S[10,1]?	◯		◯							

$\min_{x \in S[10,2]} A[x] = 1$

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	3	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
∈S[10,2]?		◯							◯	

Example 1

$\min_{x \in S[10,3]} A[x] = 4$

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	3	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
∈S[10,3]?				◯		◯				

$\min_{x \in S[10,4]} A[x] = 6$

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	3	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
∈S[10,4]?					◯					

Example 1

$\min_{x \in S[10,5]} A[x] = 8$

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	3	0	5	6	4	9	8	1	7
L[i]	1	2	1	3	4	3	5	5	2	5
∈S[10,5]?							◯	◯		

We have that $\min_{x \in S[10,5]} A[x] = 8 \geq 7 = A[10]$, so no increasing subsequence of length 6 ends at A[10].

Example 2

$\min_{x \in S[10,1]} A[x] = 0$

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	1	3
∈S[10,1]?	◯		◯						◯	

$\min_{x \in S[10,2]} A[x] = 3$

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	1	3
∈S[10,2]?		◯		◯	◯		◯			

Example 2

$\min_{x \in S[10,3]} A[x] = 7$

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	1	3
$\in$S[10,3]?						◯				

We have that $\min_{x \in S[10,3]} A[x] = 7 \geq 4 = A[10]$, so no increasing subsequence of length 4 ends at A[10].

More efficient method

$\min_{x\in\emptyset}A[x]=\infty$

- Let $B[k,p]=\min_{x\in S[k,p]}A[x]$
- $\max_{j<k,A[j]<A[k]}L[j]=\max\{p:B[k,p]<A[k]\}$
  - O(logn): $B[k,1]\le B[k,2]\le\ldots\le B[k,k-1]$.
- There is at most one p such that $B[k,p]\neq B[k+1,p]$ (why?)
  - Use this property to maintain an array B'[1..n] such that B'[p]=B[k,p] when computing L[k].

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2									
B[k,2]	∞									
B[k,3]	∞									
B[k,4]	∞									
B[k,5]	∞									

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2								
B[k,2]	∞	9								
B[k,3]	∞	∞								
B[k,4]	∞	∞								
B[k,5]	∞	∞								

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0							
B[k,2]	∞	9	9							
B[k,3]	∞	∞	∞							
B[k,4]	∞	∞	∞							
B[k,5]	∞	∞	∞							

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0						
B[k,2]	∞	9	9	8						
B[k,3]	∞	∞	∞	∞						
B[k,4]	∞	∞	∞	∞						
B[k,5]	∞	∞	∞	∞						

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0					
B[k,2]	∞	9	9	8	6					
B[k,3]	∞	∞	∞	∞	∞					
B[k,4]	∞	∞	∞	∞	∞					
B[k,5]	∞	∞	∞	∞	∞					

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0	0				
B[k,2]	∞	9	9	8	6	6				
B[k,3]	∞	∞	∞	∞	∞	7				
B[k,4]	∞	∞	∞	∞	∞	∞				
B[k,5]	∞	∞	∞	∞	∞	∞				

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0	0	0			
B[k,2]	∞	9	9	8	6	6	3			
B[k,3]	∞	∞	∞	∞	∞	7	7			
B[k,4]	∞	∞	∞	∞	∞	∞	∞			
B[k,5]	∞	∞	∞	∞	∞	∞	∞			

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0	0	0	0		
B[k,2]	∞	9	9	8	6	6	3	3		
B[k,3]	∞	∞	∞	∞	∞	7	7	7		
B[k,4]	∞	∞	∞	∞	∞	∞	∞	9		
B[k,5]	∞	∞	∞	∞	∞	∞	∞	∞		

Execution on example 2

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

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0	0	0	0	0	
B[k,2]	∞	9	9	8	6	6	3	3	3	
B[k,3]	∞	∞	∞	∞	∞	7	7	7	7	
B[k,4]	∞	∞	∞	∞	∞	∞	∞	9	9	
B[k,5]	∞	∞	∞	∞	∞	∞	∞	∞	∞	

Execution on example 2

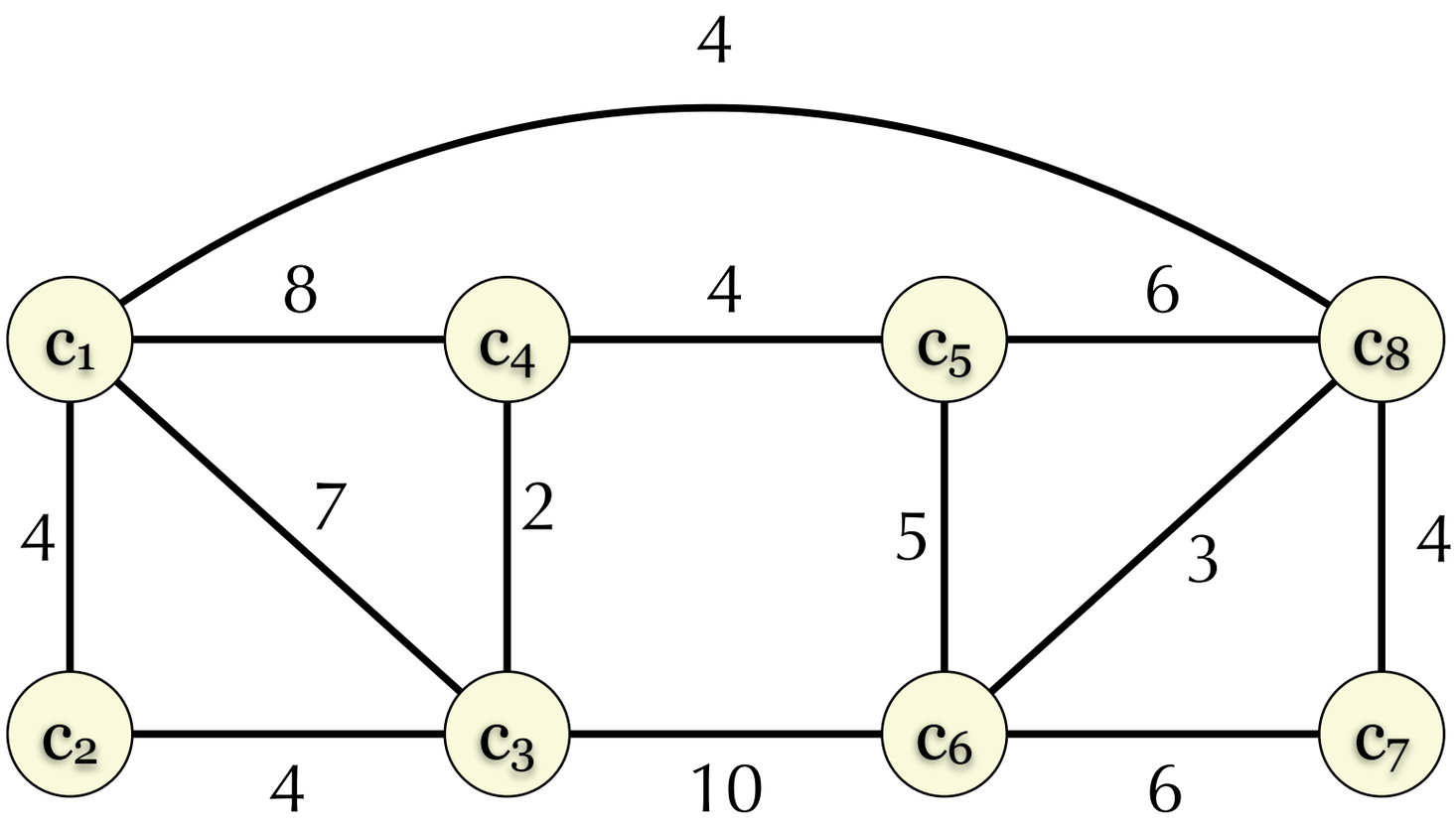
k	1	2	3	4	5	6	7	8	9	10
A[k]	2	9	0	8	6	7	3	9	0	4
L[k]	1	2	1	2	2	3	2	4	1	3

k	1	2	3	4	5	6	7	8	9	10
B[k,1]	2	2	0	0	0	0	0	0	0	0
B[k,2]	∞	9	9	8	6	6	3	3	3	3
B[k,3]	∞	∞	∞	∞	∞	7	7	7	7	4
B[k,4]	∞	∞	∞	∞	∞	∞	∞	9	9	9
B[k,5]	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

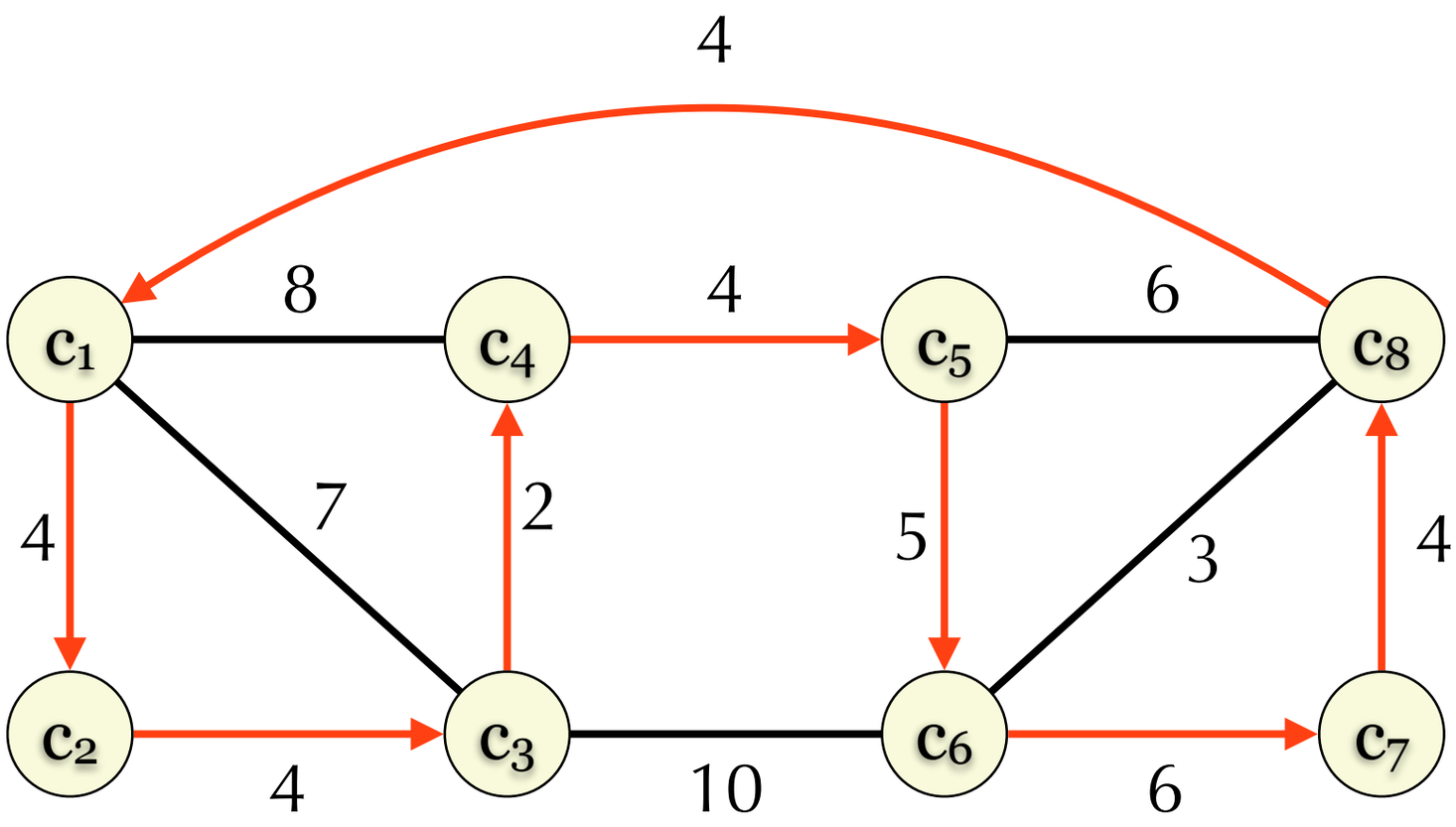
Traveling Salesperson

- A salesperson has to sell products to n cities c_1, …, c_n.
  - Starts and ends at c_1.
- Traveling cost from c_i to c_j: $w_{i,j}$
- How to minimize the traveling cost?
- Complete search: O(n!)
- How to solve the problem for n=16?
  - Dynamic programming: $O(n^2 2^n)$

Example



Example



Recursive

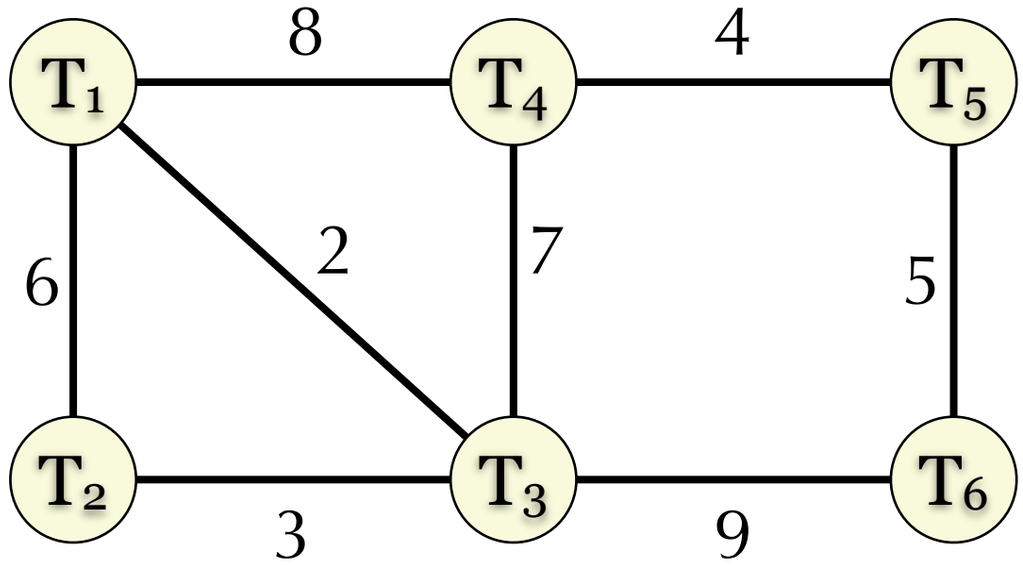
- How to define a proper recursion?
  - The hardcore of DP!
- F(s,C): The minimum cost to extend a path, which passes all cities in C and ends at s, into a proper tour.
- Answer: $F(1,\{c_1\})$
- The easy case: $F(s,\{c_1,\ldots,c_n\})=w_{s,1}$
- The recursive part: $F(s,C)=\min_{t\notin C}F(t,C\cup\{t\})+w_{s,t}$
- Store the table: using bit set

Weighted Perfect Matching

- There are 2n items $T_1,\ldots,T_{2n}$ for sale.
- Discount policy: You'll get c(i,j) discount if you buy T_i and T_j simultaneously.
- How to achieve the maximum total discount?
- Brute-force: verify all possible matchings
- DP: prepare a table to store all optimal solutions to all subproblems.

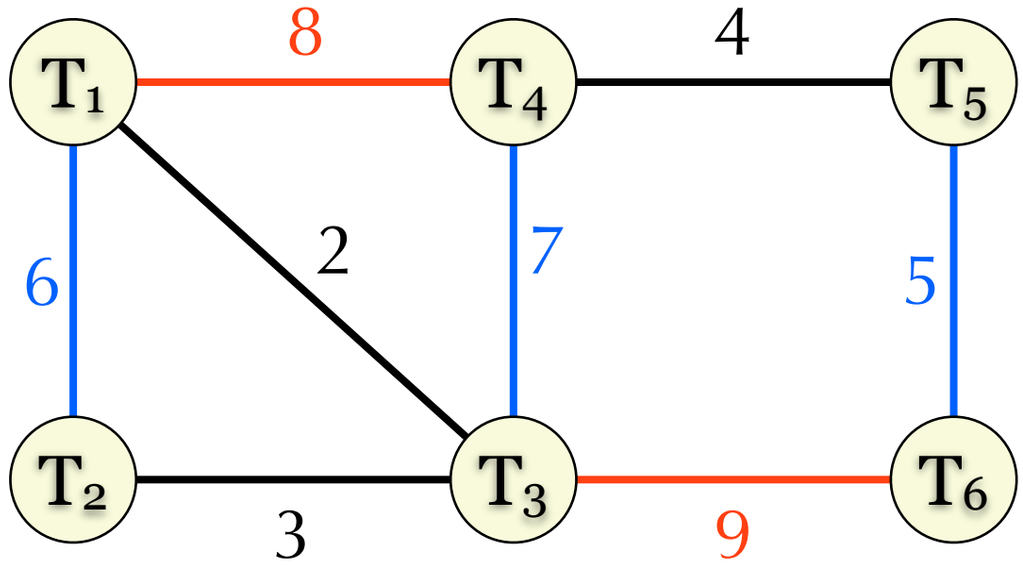
Example

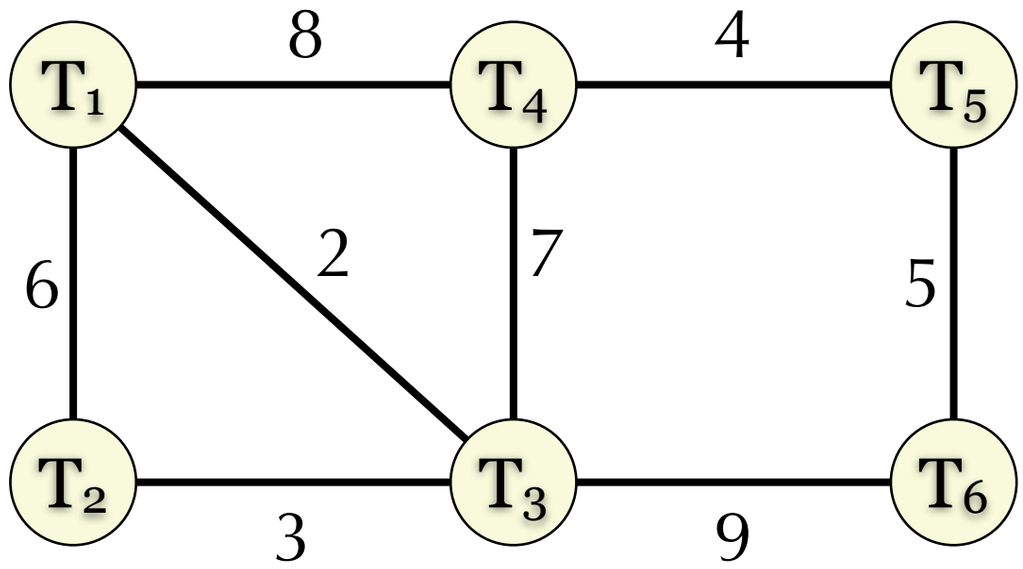
Edge: discount



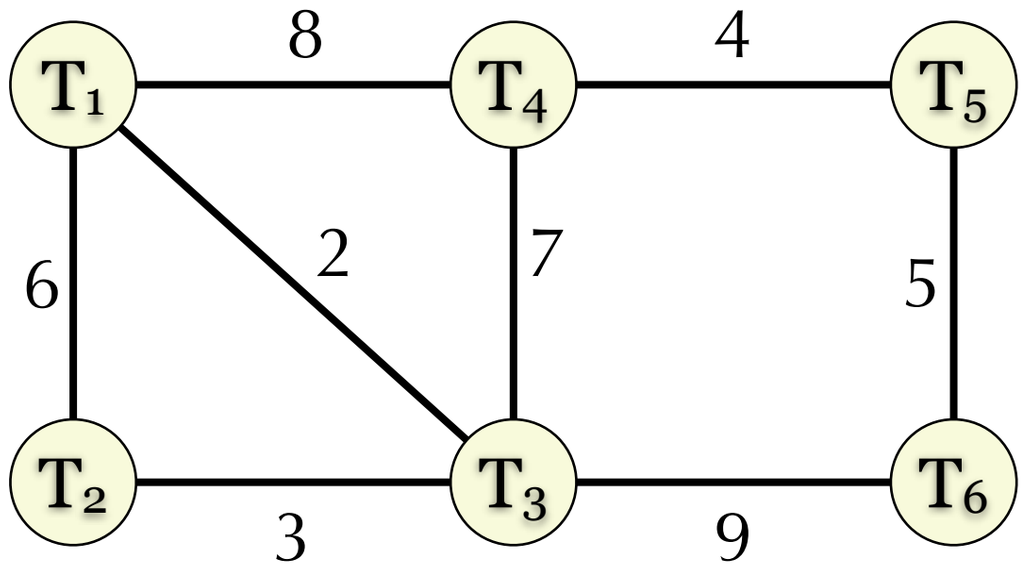
Greedy does not Work!

Edge: discount

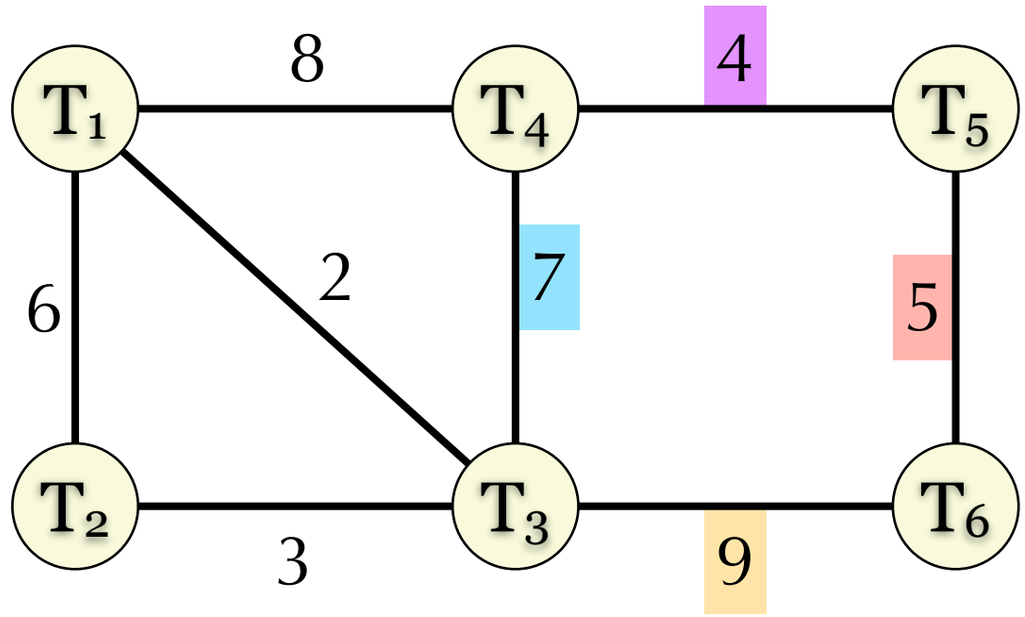




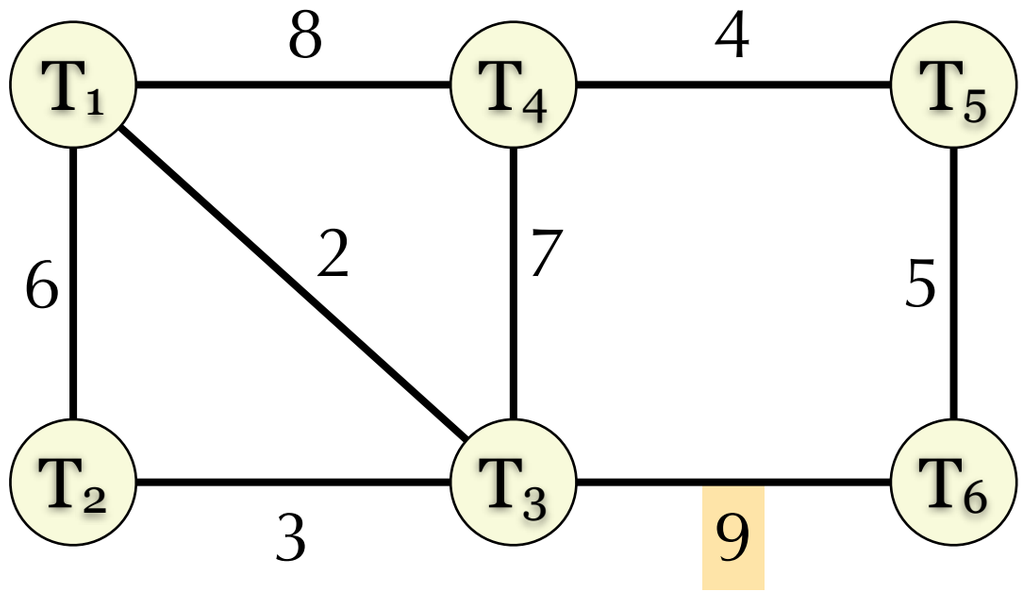
Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010		111100		011101	
000011		100010		111010		110011	
000101		001100		110110		101011	
001001		010100		101110		011011	
010001		100100		011110		100111	
100001		011000		111001		010111	
000110		101000		110101		001111	
001010		110000		101101		111111	



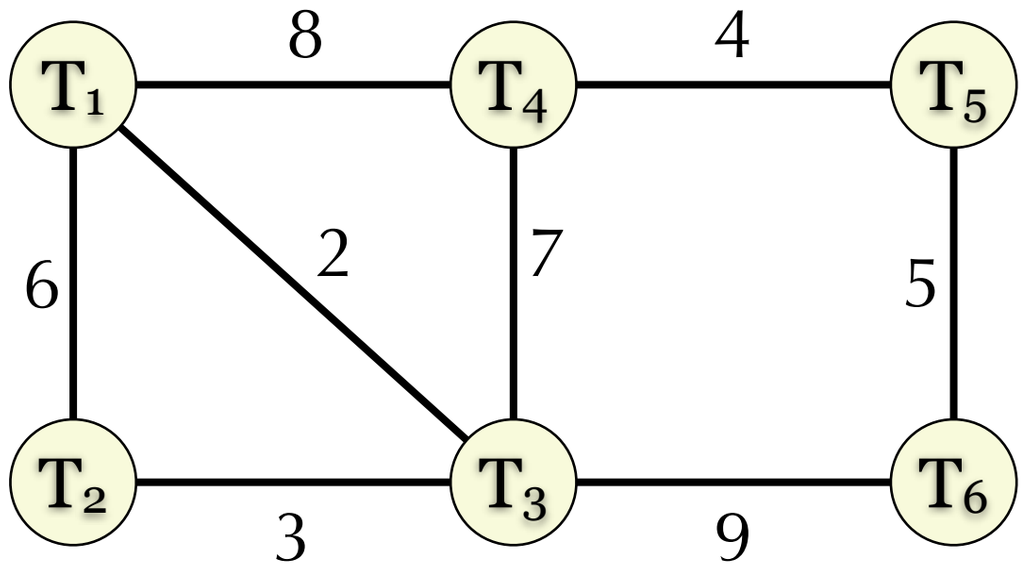
Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010		111100	5	011101	0
000011		100010		111010	0	110011	7
000101		001100		110110	9	101011	0
001001		010100		101110	0	011011	8
010001		100100		011110	0	100111	3
100001		011000		111001	4	010111	2
000110		101000		110101	0	001111	6
001010		110000		101101	0	111111	0



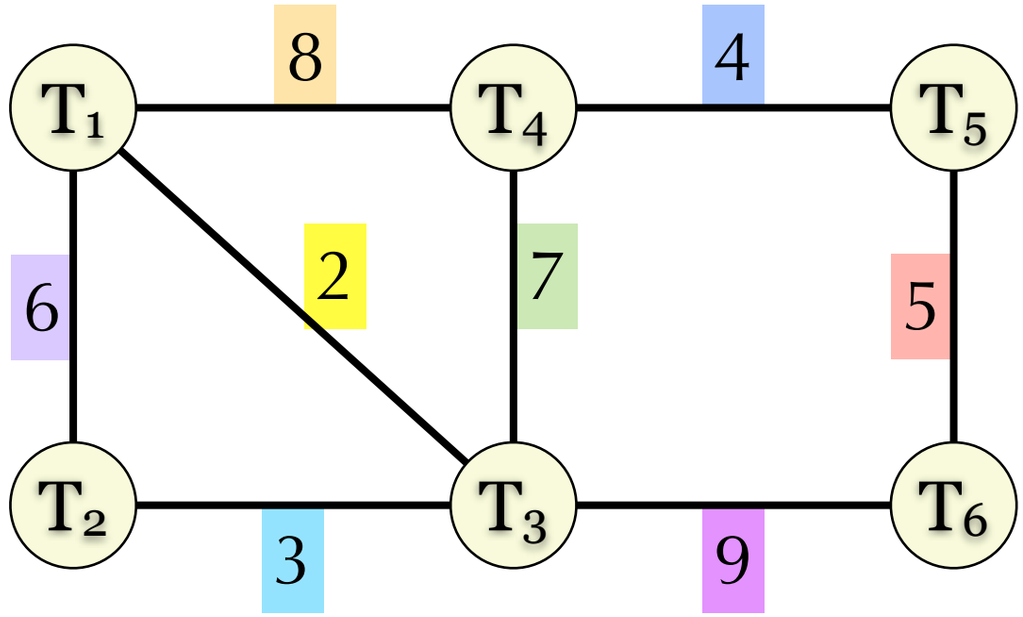
Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010		111100	5	011101	0
000011		100010		111010	0	110011	7
000101		001100		110110	9	101011	0
001001		010100		101110	0	011011	8
010001		100100		011110	0	100111	3
100001		011000		111001	4	010111	2
000110		101000		110101	0	001111	6
001010		110000		101101	0	111111	0



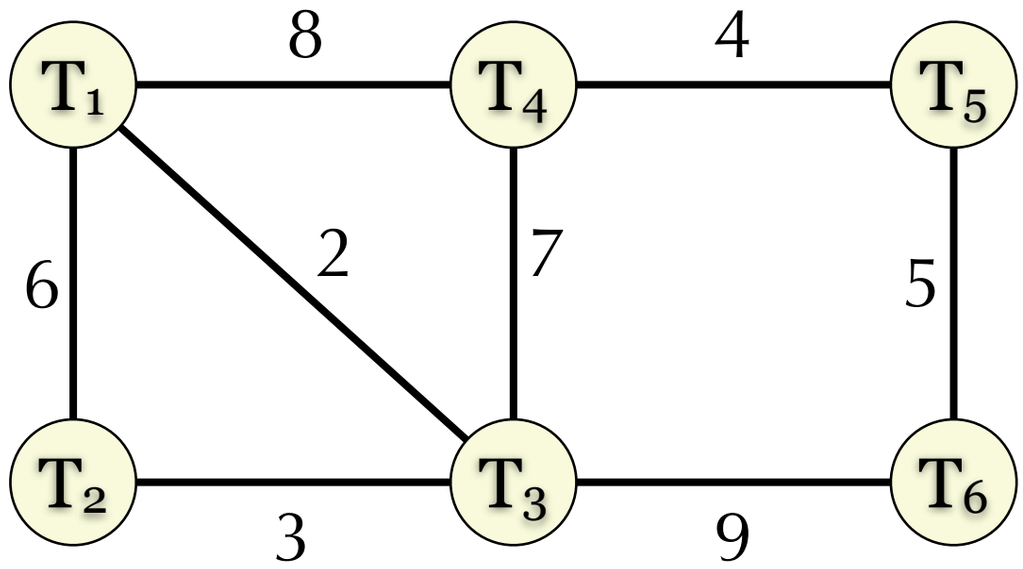
Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010		111100	5	011101	0
000011		100010		111010	0	110011	7
000101		001100		110110	9	101011	0
001001		010100		101110	0	011011	8
010001		100100		011110	0	100111	3
100001		011000		111001	4	010111	2
000110		101000		110101	0	001111	6
001010		110000	13	101101	0	111111	0



Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010	17	111100	5	011101	0
000011	13	100010	9	111010	0	110011	7
000101	6	001100	11	110110	9	101011	0
001001	10	010100	9	101110	0	011011	8
010001	8	100100	9	011110	0	100111	3
100001	7	011000	13	111001	4	010111	2
000110	15	101000	5	110101	0	001111	6
001010	8	110000	13	101101	0	111111	0

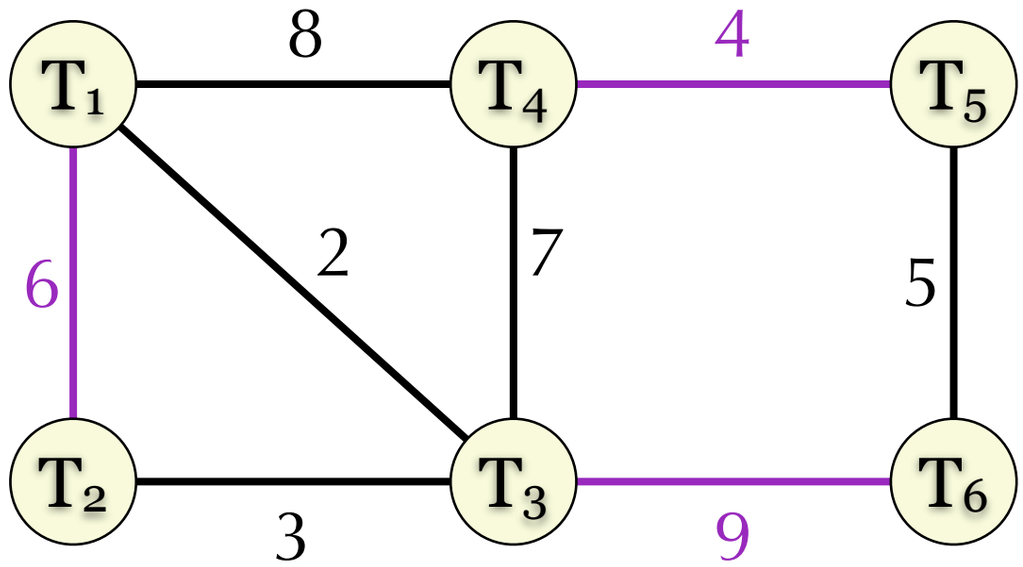


Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000		010010	17	111100	5	011101	0
000011	13	100010	9	111010	0	110011	7
000101	6	001100	11	110110	9	101011	0
001001	10	010100	9	101110	0	011011	8
010001	8	100100	9	011110	0	100111	3
100001	7	011000	13	111001	4	010111	2
000110	15	101000	5	110101	0	001111	6
001010	8	110000	13	101101	0	111111	0



Bitset	opt	Bitset	opt	Bitset	opt	Bitset	opt
000000	19	010010	17	111100	5	011101	0
000011	13	100010	9	111010	0	110011	7
000101	6	001100	11	110110	9	101011	0
001001	10	010100	9	101110	0	011011	8
010001	8	100100	9	011110	0	100111	3
100001	7	011000	13	111001	4	010111	2
000110	15	101000	5	110101	0	001111	6
001010	8	110000	13	101101	0	111111	0

Optimal Solution



Subset Sum

- Given a set of integers $S=\{x_1,\ldots,x_n\}$ where $x_1,\ldots,x_n \in [1,W]$ and an integer $y \in [1,nW]$, is there a subset $S' \subseteq S$ such that $\Sigma_{x \in S'} x = y$?
- Examples:
  - $S=\{1,2,5,8\}$ and $y=4$: No.
  - $S=\{1,2,3,4\}$ and $y=5$: Yes, $5=1+4=2+3$
- Brute force: try all 2^n subsets… $n \leq 30$
- Dynamic programming: $O(ny)$
  - Works for small W

Subset Sum

S={1,3,4,6,6,9}, y=9

i	0	1	2	3	4	5	6	7	8	9
x_i	◯									
1										
3										
4										
6										
6										
9										

Subset Sum

S={1,3,4,6,6,9}, y=9

i	0	1	2	3	4	5	6	7	8	9
x_i	○									
1	○	○								
3	○	○		○	○					
4	○	○		○	○	○		○	○	
6	○	○		○	○	○	○	○	○	○
6	○	○		○	○	○	○	○	○	○
9	○	○		○	○	○	○	○	○	○

Subset Sum

S={2,2,4,4,6,8}, y=9

i	0	1	2	3	4	5	6	7	8	9
x_i	○									
2	○		○							
2	○		○		○					
4	○		○		○		○		○	
4	○		○		○		○		○	
6	○		○		○		○		○	
8	○		○		○		○		○	

Space Compression

S={1,3,4,6,6,9}, y=9

i	0	1	2	3	4	5	6	7	8	9
dp[i]	◯									

i	0	1	2	3	4	5	6	7	8	9
dp[i]	◯	◯								

1

i	0	1	2	3	4	5	6	7	8	9
dp[i]	◯	◯		◯	◯					

3

Space Compression

S={1,3,4,6,6,9}, y=9

i	0	1	2	3	4	5	6	7	8	9
dp[i]	○	○		○	○					

i	0	1	2	3	4	5	6	7	8	9	
dp[i]	○	○		○	○	○		○	○		4

i	0	1	2	3	4	5	6	7	8	9	
dp[i]	○	○		○	○	○	○	○	○	○	6