Design and Analysis of Algorithms, MTech-I (1st semester)

Chapter 6: NP Theory - I

October 7, 2022

Devesh C Jinwala,

Professor in CSE, SVNIT, Surat and Adjunct Professor, IITJammu & Dean (R&C), SVNIT

Department of Computer Science and Engineering, SVNIT, Surat

Broad Contents of the talks

- Talk1: Complexity Classes of Problems and a few "Hard" problems.

Broad Contents of the talks

- Talk1: Complexity Classes of Problems and a few "Hard" problems.
- Talk2: Understanding and Working with Problem Reductions.

Broad Contents of the talks

- Talk1: Complexity Classes of Problems and a few "Hard" problems.
- Talk2: Understanding and Working with Problem Reductions.
- Talk3: Non-determinism, Working with NPHard, NPComplete.

Topics to be discussed

A. **Talk1: Complexity Classes of Problems and a few "Hard" problems.**
   1. What does solving problems algorithmically, mean?
   2. Classifying problems
   3. A Motivating Example to illustrate hardness
   4. Some Hard Problems

Topics to be discussed...

- **B** **Talk2: Relating Problem Hardness & Polynomial Reductions**
  1. Reductions
  2. How can we relate hardness of two problems ?
  3. How can we relate solvability of two problems ?
  4. Polynomial Reduction of one problem to the other
  5. Polynomial Equivalence of one problem to the other
  6. Three methods of reductions: illustrations

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems
  5. Proofs associated

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems
  5. Proofs associated
  6. Summarizing

Why do we need an algorithm?

- Is the software development, in general, an algorithmic activity or otherwise ?
- Two types of problems solving approaches
  - Algorithmic. What is the advantage ? Is it required to be iterative ?

Why do we need an algorithm?

- Is the software development, in general, an algorithmic activity or otherwise ?
- Two types of problems solving approaches
  - Algorithmic. What is the advantage ? Is it required to be iterative ?
  - Intuitive. Then, has to be iterative.

Why do we need an algorithm?...

- How can we solve the following problem?

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size n of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size n of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51

- How many comparisons your program makes ? How many iterations of the for loop ?

Why do we need an algorithm?...

- How can we solve the following problem?

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size n of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51
32	22	42	62	102	92	72	82	12	52
33	23	43	63	103	93	73	83	13	53
34	24	44	64	104	94	74	84	14	54
35	25	45	65	105	95	75	85	15	55
36	26	46	66	106	96	76	86	16	56
37	27	47	67	107	97	77	87	17	57

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size n of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51
32	22	42	62	102	92	72	82	12	52
33	23	43	63	103	93	73	83	13	53
34	24	44	64	104	94	74	84	14	54
35	25	45	65	105	95	75	85	15	55
36	26	46	66	106	96	76	86	16	56
37	27	47	67	107	97	77	87	17	57

- How many comparisons your program makes ? How many iterations of the for loop ?

Why do we need an algorithm?...

- How can we solve the following problem?

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size 10^{1000} of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51
32	22	42	62	102	92	72	82	12	52
33	23	43	63	103	93	73	83	13	53
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
34	24	44	64	104	94	74	84	14	54
35	25	45	65	105	95	75	85	15	55
36	26	46	66	106	96	76	86	16	56
37	27	47	67	107	97	77	87	17	57

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size 10^{1000} of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51
32	22	42	62	102	92	72	82	12	52
33	23	43	63	103	93	73	83	13	53
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
34	24	44	64	104	94	74	84	14	54
35	25	45	65	105	95	75	85	15	55
36	26	46	66	106	96	76	86	16	56
37	27	47	67	107	97	77	87	17	57

- How many comparisons your program makes ? How many iterations of the for loop ?

Why do we need an algorithm?...

- How can we solve the following problem?
- Given a vector of size 10^{1000} of integer data objects, rearrange the vector in the ascending order of the values of the data objects.

31	21	41	61	101	91	71	81	11	51
32	22	42	62	102	92	72	82	12	52
33	23	43	63	103	93	73	83	13	53
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
34	24	44	64	104	94	74	84	14	54
35	25	45	65	105	95	75	85	15	55
36	26	46	66	106	96	76	86	16	56
37	27	47	67	107	97	77	87	17	57

- How many comparisons your program makes ? How many iterations of the for loop ?
- So, now why do we need an algorithm ?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?
- Design an algorithm ?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?
- Design an algorithm ?
- Proving the correctness of an algorithm

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?
- Design an algorithm ?
- Proving the correctness of an algorithm
- Exponential complexity?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?
- Design an algorithm ?
- Proving the correctness of an algorithm
- Exponential complexity?
- Polynomial Time complexity ? How to define it ?

Approach to solve a problem algorithmically

- Given a computational problem to solve, how would you attempt to solve it ?
- Brute-force ?
- Design an algorithm ?
- Proving the correctness of an algorithm
- Exponential complexity?
- Polynomial Time complexity ? How to define it ?
- Explore the ways to improve?

Polynomial or Intractable ?

	n	$n \lg n$	N^2	N^3	1.5^n	2^n	$n!$
n=10	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 4 sec
n=30	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} yrs
n=50	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 yrs	very long
n=100	< 1 sec	< 1 sec	< 1 sec	1 sec	12.89 yrs	10^{17} yrs	< 4 sec
n=1000	< 1 sec	< 1 sec	1 sec	18 min sec	very long	very long	very long
n=10K	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
n=100K	< 1 sec	2 sec	3 hrs	32 yrs	very long	very long	very long
n=1M	1 sec	20 sec	12 days	31.71 yrs	very long	very long	very long

Figure: Complexity Orders related to the time of execution

Problems you must have studied so far

- Sorting

Problems you must have studied so far

- Sorting
- Searching

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph
- Primality testing

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph
- Primality testing
- Towers of Hanoi

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph
- Primality testing
- Towers of Hanoi
- Travelling Salesperson problem

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph
- Primality testing
- Towers of Hanoi
- Travelling Salesperson problem
- Program termination

Problems you must have studied so far

- Sorting
- Searching
- Minimum spanning tree
- Fractional Knapsack problem
- Shortest paths in a graph
- Primality testing
- Towers of Hanoi
- Travelling Salesperson problem
- Program termination
- ... ...

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
    - Finding maximum/minimum in an array
    - Finding the sum of n element array
    - Comparison based Sorting
    - Binary search

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem
  - Container Loading problem

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem
  - Container Loading problem
  - Activity Selection

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem
  - Container Loading problem
  - Activity Selection
  - Job scheduling to minimize average completion time

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem
  - Container Loading problem
  - Activity Selection
  - Job scheduling to minimize average completion time
  - Dijkstra and Bellman Ford SSSP Algorithms

Tutorial: Order of the Time Complexities

- What is the worst case time complexity of the algorithms to solve these problems viz.
  - Finding maximum/minimum in an array
  - Finding the sum of n element array
  - Comparison based Sorting
  - Binary search
  - Kruskal's and Prim's MST algorithms
  - Fractional Knapsack problem
  - Container Loading problem
  - Activity Selection
  - Job scheduling to minimize average completion time
  - Dijkstra and Bellman Ford SSSP Algorithms
  - Floyd Warshall's APSP Algorithms

O(n^3) highest among all

Efficient Solutions

- When can an algorithm be considered an efficient algorithm ?

Effectively Polynomial time algorithm

- A solution to a problem is generally considered to be efficient if its running time is bounded by a polynomial in input n i.e.

Efficient Solutions

- When can an algorithm be considered an efficient algorithm ?

Effectively Polynomial time algorithm

- A solution to a problem is generally considered to be efficient if its running time is bounded by a polynomial in input n i.e.
- $\exists$ a constant c, such that the running time of the algorithm $T(n)$ is $O(n^c)$, where c is some positive integer and n is the input size.

Efficient Solutions

- When can an algorithm be considered an efficient algorithm ?

Effectively Polynomial time algorithm

- A solution to a problem is generally considered to be efficient if its running time is bounded by a polynomial in input n i.e.
- $\exists$ a constant c, such that the running time of the algorithm $T(n)$ is $O(n^c)$, where c is some positive integer and n is the input size.
- c could be either constant degree of – n^2, n^3, n^4 ... OR $n\ log\ n$

Efficient Solutions

- When can an algorithm be considered an efficient algorithm ?

Effectively Polynomial time algorithm

- A solution to a problem is generally considered to be efficient if its running time is bounded by a polynomial in input n i.e.
- $\exists$ a constant c, such that the running time of the algorithm $T(n)$ is $O(n^c)$, where c is some positive integer and n is the input size.
- c could be either constant degree of – n^2, n^3, n^4 ... OR $n\ log\ n$
- Searching ($O\ lg\ n$), Sorting ($O(n\ lg\ n)$), Polynomial evaluation ($O(n)$). . . . . . .

Tutorial:Why polynomial time complexity only is considered efficient?

- Three principal reasons
    - Machine Independence

Tutorial:Why polynomial time complexity only is considered efficient?

- Three principal reasons
  - Machine Independence
  - Limited to $O(n^2)$

Tutorial:Why polynomial time complexity only is considered efficient?

- Three principal reasons
  - Machine Independence
  - Limited to $O(n^2)$
  - Closure property

add two polynomial the ans is polynomial

Can every problem have an efficient solution ?

- Can every problem have an efficient i.e. one that is bounded polynomially in time, solution ?
- What then, are the non-efficient (i.e. inefficient!) solutions?

Non-efficient solutions

- A solution to a problem is generally considered to be non-efficient if it is found using the brute force approach.

Can every problem have an efficient solution ?

- Can every problem have an efficient i.e. one that is bounded polynomially in time, solution ?
- What then, are the non-efficient (i.e. inefficient!) solutions?

Non-efficient solutions

- A solution to a problem is generally considered to be non-efficient if it is found using the brute force approach.
- No $O(n^k)$ algorithm is known

Can every problem have an efficient solution ?

- Can every problem have an efficient i.e. one that is bounded polynomially in time, solution ?
- What then, are the non-efficient (i.e. inefficient!) solutions?

Non-efficient solutions

- A solution to a problem is generally considered to be non-efficient if it is found using the brute force approach.
- No $O(n^k)$ algorithm is known
- That is, time required is $O(n^n)$ or $O(n!)$ or $O(k^n)$ time required....

Can every problem have an efficient solution ?

- Can every problem have an efficient i.e. one that is bounded polynomially in time, solution ?
- What then, are the non-efficient (i.e. inefficient!) solutions?

Non-efficient solutions

- A solution to a problem is generally considered to be non-efficient if it is found using the brute force approach.
- No $O(n^k)$ algorithm is known
- That is, time required is $O(n^n)$ or $O(n!)$ or $O(k^n)$ time required....
- Traveling Salesperson takes $(O(n^2 2^n))$, 0/1 Knapsack, takes $(O(2^{n/2}))$ . . . .

Can every problem have an efficient solution ?

- Can every problem have an efficient i.e. one that is bounded polynomially in time, solution ?
- What then, are the non-efficient (i.e. inefficient!) solutions?

Non-efficient solutions

- A solution to a problem is generally considered to be non-efficient if it is found using the brute force approach.
- No $O(n^k)$ algorithm is known
- That is, time required is $O(n^n)$ or $O(n!)$ or $O(k^n)$ time required....
- Traveling Salesperson takes $(O(n^2 2^n))$, 0/1 Knapsack, takes $(O(2^{n/2}))$ . . . .
- Though this may not be true always. . . . .

Inefficient Solutions

- Why are inefficient solutions are those that are obtained using the brute-force ?

Non-efficient solutions

- Bruteforce algorithms search exhaustively through the entire solution space i.e. . . . . ???

Inefficient Solutions

- Why are inefficient solutions are those that are obtained using the brute-force ?

Non-efficient solutions

- Bruteforce algorithms search exhaustively through the entire solution space i.e. . . . . ???
- e.g. how do we carry out selection using brute force ?

Inefficient Solutions

- Why are inefficient solutions are those that are obtained using the brute-force ?

Non-efficient solutions

- Bruteforce algorithms search exhaustively through the entire solution space i.e. . . . . ???
- e.g. how do we carry out selection using brute force ?
- e.g. how do we carry out sorting using brute force ?

Inefficient Solutions

- Why are inefficient solutions are those that are obtained using the brute-force ?

Non-efficient solutions

- Bruteforce algorithms search exhaustively through the entire solution space i.e. . . . . ???
- e.g. how do we carry out selection using brute force ?
- e.g. how do we carry out sorting using brute force ?
- complexities of sorting – comparison based, non-comparison based, non-comparison based brute force based... ... ...

Inefficient Solutions

- Why are inefficient solutions are those that are obtained using the brute-force ?

Non-efficient solutions

- Bruteforce algorithms search exhaustively through the entire solution space i.e. . . . . ???
- e.g. how do we carry out selection using brute force ?
- e.g. how do we carry out sorting using brute force ?
- complexities of sorting – comparison based, non-comparison based, non-comparison based brute force based... ... ...
- If the input size is n, typical time taken is 2^n.

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?
- Decidable or Undecidable problems

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?
- Decidable or Undecidable problems
- Decidable problems

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?
- Decidable or Undecidable problems
- Decidable problems
    - Tractable problems

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?
- Decidable or Undecidable problems
- Decidable problems
  - Tractable problems
  - Intractable problems

Desiderata: Classifying Problems

Why do we wish to classify the problems?

- What if we encounter a new problem tomorrow?
- Decidable or Undecidable problems
- Decidable problems
  - Tractable problems
  - Intractable problems
- In 1936, Alan Turing proved that the halting problem - the question of whether or not a Turing machine halts on a given program - is undecidable. This result was later generalized by Rice's theorem.

Should we waste our time designing an algorithm to a problem that is globally and over time believed to be undecidable OR intractable ?

A Motivating Example

A Motivating Example

Machine scheduling to minimize the **average job completion time**

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots \ldots\ldots m_n$ such that

A Motivating Example

Machine scheduling to minimize the **average job completion time**

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots\ldots m_n$ such that
  - no machine processes more than one process at a time

A Motivating Example

Machine scheduling to minimize the **average job completion time**

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots m_n$ such that
  - no machine processes more than one process at a time
  - no process is executed by more than one machine

A Motivating Example

Machine scheduling to minimize the **average job completion time**

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots \ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots \ldots m_n$ such that
  - no machine processes more than one process at a time
  - no process is executed by more than one machine
  - there is non-preemptive scheduling

A Motivating Example

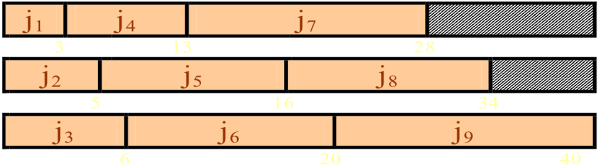
Machine scheduling to minimize the **average job completion time**

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots\ldots m_n$ such that
    - no machine processes more than one process at a time
    - no process is executed by more than one machine
    - there is non-preemptive scheduling
    - the average job completion time is minimized.

sortest job first algorithm

A Motivating Example...

Goal: Minimizing the average job completion time.....



M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
3	5	6	10	11	14	15	18	20
3	5	6	13	16	20	28	34	40

Figure: What is the Average job completion time?

A Motivating Example...

Goal: To prove that the SJF (or SRTN, if premptive scheduling) scheduling indeed ensures optimal i.e. minimal average job completion time.....
Proof:

A Motivating Example...

Machine scheduling to minimize the final completion time of machines used.

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots\ldots m_n$ such that

A Motivating Example...

Machine scheduling to minimize the final completion time of machines used.

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots \ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots \ldots m_n$ such that
  - no machine processes more than one process at a time

A Motivating Example...

Machine scheduling to minimize the final completion time of machines used.

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots m_n$ such that
  - no machine processes more than one process at a time
  - no process is executed by more than one machine

A Motivating Example...

Machine scheduling to minimize the final completion time of machines used.

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots\ldots m_n$ such that
  - no machine processes more than one process at a time
  - no process is executed by more than one machine
  - there is non-preemptive scheduling

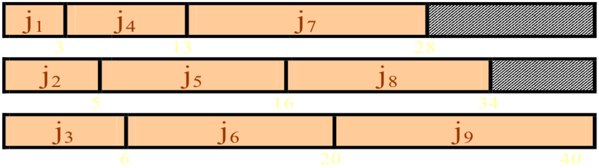
A Motivating Example...

Machine scheduling to minimize the final completion time of machines used.

- Given a set of m processes $j_1, j_2, j_3, \ldots j_m$ with running times $t_1, t_2, t_3 \ldots\ldots t_t$ to be scheduled on specified n no of machines $m_1, m_2, m_3, \ldots\ldots m_n$ such that
  - no machine processes more than one process at a time
  - no process is executed by more than one machine
  - there is non-preemptive scheduling
  - the final completion time is minimized.

A Motivating Example...

Goal: Minimizing the Final Completion time of processors.....



M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
3	5	6	10	11	14	15	18	20
3	5	6	13	16	20	28	34	40

Figure: What is the Final Completion time in this schedule?

A Motivating Example...

Goal: Minimizing the Final Completion time of processors.....

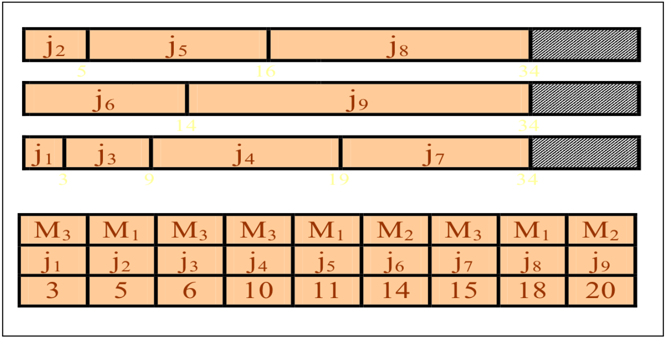


Figure: What is the Final Completion time in this schedule?

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \ldots \ldots \ldots \ldots j_m$ with running times $t_1, t_2, t_3 \ldots \ldots \ldots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \dots \dots \dots \dots j_m$ with running times $t_1, t_2, t_3 \dots \dots \dots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.
- Consider that as an example, we have the jobs $j_1, j_2, j_3 \dots \dots \dots j_7$ with running times 2,100,2,100,2,100,2

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \dots \dots \dots \dots j_m$ with running times $t_1, t_2, t_3 \dots \dots \dots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.
- Consider that as an example, we have the jobs $j_1, j_2, j_3 \dots \dots \dots j_7$ with running times 2,100,2,100,2,100,2
- How do we schedule these jobs on P_1 and P_2 ?

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \dots \dots \dots \dots j_m$ with running times $t_1, t_2, t_3 \dots \dots \dots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.
- Consider that as an example, we have the jobs $j_1, j_2, j_3 \dots \dots \dots j_7$ with running times 2,100,2,100,2,100,2
- How do we schedule these jobs on P_1 and P_2 ?
- Say we schedule all the ODD jobs on P_1 and the even jobs on P_2, then, how are the first two jobs scheduled ?

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \ldots \ldots \ldots \ldots j_m$ with running times $t_1, t_2, t_3 \ldots \ldots \ldots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.
- Consider that as an example, we have the jobs $j_1, j_2, j_3 \ldots \ldots \ldots j_7$ with running times 2,100,2,100,2,100,2
- How do we schedule these jobs on P_1 and P_2 ?
- Say we schedule all the ODD jobs on P_1 and the even jobs on P_2, then, how are the first two jobs scheduled ?
- Then, the last job is ______ & finishes at time ______.

Motivating Example: Attempt#1

- Given a set of m processes $j_1, j_2, j_3, \ldots \ldots \ldots \ldots j_m$ with running times $t_1, t_2, t_3 \ldots \ldots \ldots t_t$ to be scheduled on say 2 no of machines m_1, m_2 such as to minimize the final completion time that is to ensure that last job finishes earliest.
- Consider that as an example, we have the jobs $j_1, j_2, j_3 \ldots \ldots \ldots j_7$ with running times 2,100,2,100,2,100,2
- How do we schedule these jobs on P_1 and P_2 ?
- Say we schedule all the ODD jobs on P_1 and the even jobs on P_2, then, how are the first two jobs scheduled ?
- Then, the last job is ______ & finishes at time ______.
- We wish to improve upon this finish time of whatever is the last job to execute i.e. we wish to evenly distribute the jobs across the two processors s.t. . . . .

Motivating Example: Attempt#2

- For the given job mix, put the first job on P_1, second job on P_2 i.e.

Motivating Example: Attempt#2

- For the given job mix, put the first job on P_1, second job on P_2 i.e.
- i.e.

P_1	P_2
j_1	j_2

Motivating Example: Attempt#2

- For the given job mix, put the first job on P_1, second job on P_2 i.e.
- i.e.

P_1	P_2
j_1	j_2

- When scheduling j_3, schedule it on the least lightly loaded processor i.e.

Motivating Example: Attempt#2

- For the given job mix, put the first job on P_1, second job on P_2 i.e.
- i.e.

P_1	P_2
j_1	j_2

- When scheduling j_3, schedule it on the least lightly loaded processor i.e.
- Check if $t_1 > t_2$, if so put j_3 on P_2 otherwise on P_1.

Motivating Example: Attempt#2

- For the given job mix, put the first job on P_1, second job on P_2 i.e.
- i.e.

P_1	P_2
j_1	j_2

- When scheduling j_3, schedule it on the least lightly loaded processor i.e.
- Check if $t_1 > t_2$, if so put j_3 on P_2 otherwise on P_1.
- What would be the schedule in the previous example with this approach i.e. for the job mix with the jobs j_1, j_2, j_3 ... ... j_7 with running times 2,100,2,100,2,100,2 ?

Motivating Example: Attempt#2...

- If we put the first job on P_1, second job on P_2 for the job mix with the jobs j_1, j_2, j_3 ... ... j_7 with running times 2,100,2,100,2,100,2, what is the schedule ?

P_1	P_2	Comment
$j_1(2)$	$j_2(100)$	
$j_3(2)$		$t_1 < t_2$
$j_4(100)$		$t_1 + t_3 < t_2$
	$j_5(2)$	$t_2 < t_1 + t_3 + t_4$
	$j_6(100)$	$t_2 + t_5 < t_1 + t_3 + t_4$
$j_7(2)$		... ...

Motivating Example: Attempt#2...

- If we put the first job on P_1, second job on P_2 for the job mix with the jobs j_1, j_2, j_3 ... ... j_7 with running times 2,100,2,100,2,100,2, what is the schedule ?

P_1	P_2	Comment
$j_1(2)$	$j_2(100)$	
$j_3(2)$		$t_1 < t_2$
$j_4(100)$		$t_1 + t_3 < t_2$
	$j_5(2)$	$t_2 < t_1 + t_3 + t_4$
	$j_6(100)$	$t_2 + t_5 < t_1 + t_3 + t_4$
$j_7(2)$		... ...

- What is the earliest finish time now?

Motivating Example: Attempt#2...

- If we put the first job on P_1, second job on P_2 for the job mix with the jobs j_1, j_2, j_3 ... ... j_7 with running times 2,100,2,100,2,100,2, what is the schedule ?

P_1	P_2	Comment
$j_1(2)$	$j_2(100)$	
$j_3(2)$		$t_1 < t_2$
$j_4(100)$		$t_1 + t_3 < t_2$
	$j_5(2)$	$t_2 < t_1 + t_3 + t_4$
	$j_6(100)$	$t_2 + t_5 < t_1 + t_3 + t_4$
$j_7(2)$		... ...

- What is the earliest finish time now?
- Is it an improvement over the previous attempt?

Motivating Example: Attempt#2...

- However, will this work always ?

Motivating Example: Attempt#2...

- However, will this work always ?
- Think of a counterexample that shows it doesn't work. . . .

Motivating Example: Attempt#3

- Sort the jobs in ascending order of their execution times. . . . and then

Motivating Example: Attempt#3

- Sort the jobs in ascending order of their execution times. . . . and then
- follow the same approach as before i.e. for the given job mix, put the first job on P_1, second job on P_2 i.e.

Motivating Example: Attempt#3

- Sort the jobs in ascending order of their execution times.... and then
- follow the same approach as before i.e. for the given job mix, put the first job on P_1, second job on P_2 i.e.
  - When scheduling j3, schedule it on the least lightly loaded processor i.e.

Motivating Example: Attempt#3

- Sort the jobs in ascending order of their execution times.... and then
- follow the same approach as before i.e. for the given job mix, put the first job on P_1, second job on P_2 i.e.
  - When scheduling j3, schedule it on the least lightly loaded processor i.e.
  - Check if $t_1 > t_2$, if so put j_3 on P_2 otherwise on P_1.

Motivating Example: Attempt#3

- Sort the jobs in ascending order of their execution times.... and then
- follow the same approach as before i.e. for the given job mix, put the first job on P_1, second job on P_2 i.e.
  - When scheduling j3, schedule it on the least lightly loaded processor i.e.
  - Check if $t_1 > t_2$, if so put j_3 on P_2 otherwise on P_1.
  - What is the schedule that we obtain for the previous example with this approach ? What is the finish time of the last process to complete?

Motivating Example: Attempt#3...

- Given the job mix j_1, j_2, j_3 ... ., j_7 with running times 2,100,2,100,2,100,2

Motivating Example: Attempt#3...

- Given the job mix j_1, j_2, j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on

P_1	P_2	Comment

Motivating Example: Attempt#3...

- Given the job mix j_1,j_2,j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on

P_1	P_2	Comment

- What is the earliest finish time now?

Motivating Example: Attempt#3...

- Given the job mix j_1,j_2,j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on

P_1	P_2	Comment

- What is the earliest finish time now?
- Is it an improvement over the previous attempt?

Motivating Example: Attempt#3...

- Given the job mix j_1,j_2,j_3 ... ., j_7 with running times 2,100,2,100,2,100,2

Motivating Example: Attempt#3...

- Given the job mix j_1,j_2,j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on...

P_1	P_2	Comment
$j_1(2)$	$j_3(2)$	
$j_5(2)$		$t_1 <= t_3$
	$j_7(2)$	$t_1 + t_5 <= t_3$
$j_2(100)$		$t_1 + t_5 <= t_3 + t_7$
	$j_4(100)$	$t_3 + t_7 <= t_3 + t_7 < t_1 + t_5 + t_2$
$j_6(100)$		

Motivating Example: Attempt#3...

- Given the job mix j_1,j_2,j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on...

P_1	P_2	Comment
$j_1(2)$	$j_3(2)$	
$j_5(2)$		$t_1 <= t_3$
	$j_7(2)$	$t_1 + t_5 <= t_3$
$j_2(100)$		$t_1 + t_5 <= t_3 + t_7$
	$j_4(100)$	$t_3 + t_7 <= t_3 + t_7 < t_1 + t_5 + t_2$
$j_6(100)$		

- What is the earliest finish time now?

Motivating Example: Attempt#3...

- Given the job mix j_1, j_2, j_3 ... ., j_7 with running times 2,100,2,100,2,100,2
- Sort the jobs in ascending order of their execution times. . . . and then for the given job mix, put the first job on P_1, second job on P_2 and so on...

P_1	P_2	Comment
$j_1(2)$	$j_3(2)$	
$j_5(2)$		$t_1 <= t_3$
	$j_7(2)$	$t_1 + t_5 <= t_3$
$j_2(100)$		$t_1 + t_5 <= t_3 + t_7$
	$j_4(100)$	$t_3 + t_7 <= t_3 + t_7 < t_1 + t_5 + t_2$
$j_6(100)$		

- What is the earliest finish time now?
- Is it an improvement over the previous attempt?

Motivating Example: Attempt#3...

- However, will this work always ?

Motivating Example: Attempt#3...

- However, will this work always ?
- Try on the counterexample containing a job mix of j_1,j_2,j_3, j_4, j_5 with times 2,2,2,3,3

Motivating Example: Attempt#3...

- However, will this work always ?
- Try on the counterexample containing a job mix of j_1,j_2,j_3, j_4, j_5 with times 2,2,2,3,3
- What is the earliest finish time now?

Motivating Example: Attempt#3...

- However, will this work always ?
- Try on the counterexample containing a job mix of j_1,j_2,j_3, j_4, j_5 with times 2,2,2,3,3
- What is the earliest finish time now?
- Is it an improvement over the previous attempt?

Motivating Example: Counterexample: Attempt#3...

P_1	P_2	Comment
$j_1(2)$	$j_2(2)$	
$j_3(2)$		$t_1 <= t_3$
	$j_4(3)$	$t_1 + t_3 < t_2$
$j_5(3)$		$t_1 + t_3 + t_5 <= t_2 + t_4$

Figure: Attempt #3 Schedule

Motivating Example: Counterexample: Attempt#3...

P_1	P_2	Comment
$j_1(2)$	$j_2(2)$	
$j_3(2)$		$t_1 <= t_3$
	$j_4(3)$	$t_1+t_3<t_2$
$j_5(3)$		$t_1+t_3+t_5<=t_2+t_4$

Figure: Attempt #3 Schedule

P_1	P_2	Comment
$j_1(2)$	$j_4(3)$	
$j_2(2)$	$j_5(3)$	
$j_3(2)$		

Figure: Optimal Schedule

A Motivating Example: One more schedule

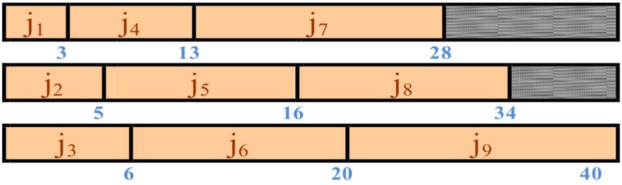
- Jobs $j_1, j_2, j_3.........j_n$ with running times $t_1, t_2, t_3,.......t_n$ to be scheduled on say three processors such as to minimize the final completion time, i.e. last job finishes the earliest...for the schedule give below

j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
3	5	6	10	11	14	15	18	20

Figure: Minimize the Final Completion time with three processors

A Motivating Example: One more schedule...

- What is the final completion time ?

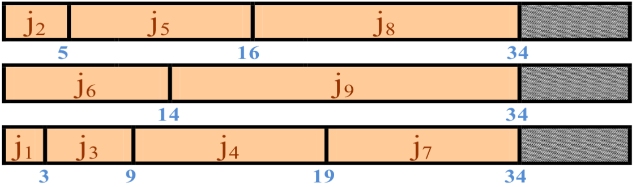


M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
3	5	6	10	11	14	15	18	20
3	5	6	13	16	20	28	34	40

Figure: Minimize the Final Completion time with three processors

A Motivating Example: One more schedule...

- What is the final completion time ?



M_3	M_1	M_3	M_3	M_1	M_2	M_3	M_1	M_2
j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9
3	5	6	10	11	14	15	18	20

Figure: Minimize the Final Completion time with three processors

- How could have this schedule been achieved ?

A Motivating Example: One more schedule...

- Applying the brute force......

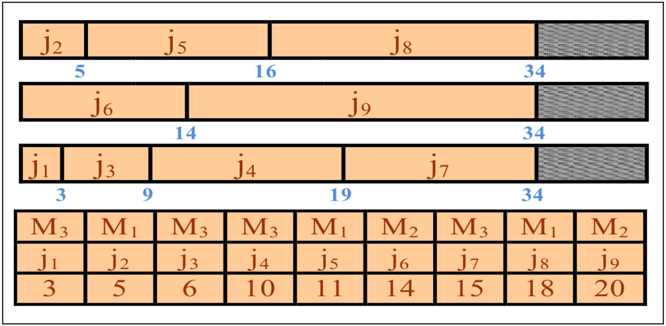


Figure: Minimize the Final Completion time with three processors

A Motivating Example: One more schedule...

- Applying the brute force......

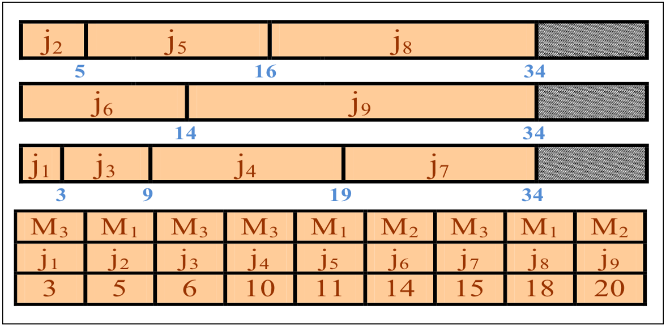


Figure: Minimize the Final Completion time with three processors

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and
- is not feasibly solvable with the existing computational power.

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and
- is not feasibly solvable with the existing computational power.

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and
- is not feasibly solvable with the existing computational power.

Why study NP Theory?

- So, now what is the rationale behind exploring the NP theory ?

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and
- is not feasibly solvable with the existing computational power.

Why study NP Theory?

- So, now what is the rationale behind exploring the NP theory ?
- The rationale is. . . . .

A Motivating Example: One more schedule...

- Well, the given problem will take exponential time complexity and
- is not feasibly solvable with the existing computational power.

Why study NP Theory?

- So, now what is the rationale behind exploring the NP theory ?
- The rationale is. . . . .
- Well, the rationale is to understand that there are problems that are going to take infeasible amount of time on a deterministic machine and so one may not waste energy . . . . . . trying to solve them. . . . . . .

Tutorial Problem:Think of one more such example

Can you think of a similar computational problem ?

Tutorial Problem:Think of one more such example

- Say there are two subjects, four teachers and there are four lectures of 1 hour each everyday.

Tutorial Problem:Think of one more such example

- Say there are two subjects, four teachers and there are four lectures of 1 hour each everyday.
- Design an algorithm to solve this problem. . . . . .

Tutorial Problem:Think of one more such example

- Say there are two subjects, four teachers and there are four lectures of 1 hour each everyday.
- Design an algorithm to solve this problem. . . . . .
- Say there are n subjects, m teachers and there are p lectures of 1 hour each everyday.

Tutorial Problem:Think of one more such example

- Say there are two subjects, four teachers and there are four lectures of 1 hour each everyday.
- Design an algorithm to solve this problem. . . . . .
- Say there are n subjects, m teachers and there are p lectures of 1 hour each everyday.
- Design an algorithm to solve this problem. . . . . .

Classifying Problems

Tractable problems = Efficiently solvable ????

Intractable problems = Inefficiently solvable ????

Figure: Classifying problems

The Frustrating news

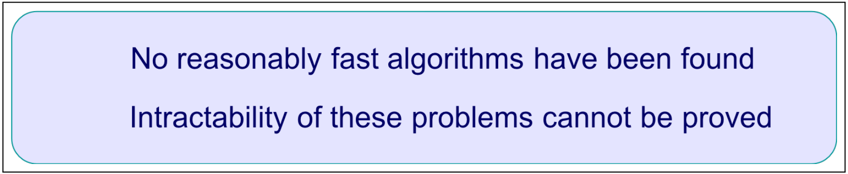


Figure: Our Focus now onwards is on Intractable problems

Why do we need to classify problems ?

- Computation has become pervasive in all walks of life a standard tool in about every academic field
  - whole subfields of Chemistry, Biology, Physics, Economics OR
  - others devoted to large-scale computational modelling, simulations and problem solving.
- We need to understand therefore, the limitations of computational power. . . .

Why do we need to classify problems ?...

- Study of P, NP theory
  - helps understand, handle various topics in allied sciences
  - helps what can be feasibly solved and what cannot be also
  - enables one to EXPLOIT the advantage due to HARDNESS of various computational problems
  - e.g. . . . .

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers
    - yes

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers
    - yes
    - no

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers
    - yes
    - no
  - because it can be proved that all algorithms take exponential time

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
    - Possible answers
        - yes
        - no
    - because it can be proved that all algorithms take exponential time
    - because it can be proved that no algorithm exists at all to solve this problem

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers
    - yes
    - no
  - because it can be proved that all algorithms take exponential time
  - because it can be proved that no algorithm exists at all to solve this problem
  - don't know

Why do we need to classify problems ?...

- Is there a polynomial-time algorithm that solves the problem?
  - Possible answers
    - yes
    - no
  - because it can be proved that all algorithms take exponential time
  - because it can be proved that no algorithm exists at all to solve this problem
  - don't know
  - don't know, but if such algorithm were to be found, then it would provide a means of solving many other problems in polynomial time

Some Hard Problems

Other interesting hard problems

- A large group of students to be grouped to work on projects so as to ensure compatibility
- Matching students in pairs . . . . . . .solvable
- Hard problems
  - Making a group of three so that each pair in each trio is compatible to each other. . . . . . ???

Other interesting hard problems...

- A large group of students to be grouped to work on projects so as to ensure compatibility
- Matching students in pairs . . . . . . .solvable
- Hard problems
  - Making a group of three so that each pair in each trio is compatible to each other. . . . . . ???
  - Finding as large group of students as possible so that each pair therein is compatible to each other. . . . . . .???

Other interesting hard problems...

- A large group of students to be grouped to work on projects so as to ensure compatibility
- Matching students in pairs ....solvable
- Hard problems
  - Making a group of three so that each pair in each trio is compatible to each other...... ??? partitioning into triangles
  - Finding as large group of students as possible so that each pair therein is compatible to each other.......??? maximum clicque
  - Wanting the student to sit across a table so that no incompatible students sit next to each other ....??? hamiltonial cycle
  - Putting students into three groups so that each student is in the same group with his/her compatible partner....??? 3 coloring problem

A few other hard problems

- Determining Hamiltonian cycle in an unweighted graph

A few other hard problems

- Determining Hamiltonian cycle in an unweighted graph
- Determining the minimal tour of cities – Travelling Salesperson problem

A few other hard problems

- Determining Hamiltonian cycle in an unweighted graph
- Determining the minimal tour of cities – Travelling Salesperson problem
- Scheduling with profits and deadlines, to maximize the profit.

A few other hard problems

- Determining Hamiltonian cycle in an unweighted graph
- Determining the minimal tour of cities – Travelling Salesperson problem
- Scheduling with profits and deadlines, to maximize the profit.
- Cliques in Social Networks - human groups form *cliques* on the basis of age, gender, race, ethnicity, religion/ideology, and many other things

A few other hard problems

- Determining Hamiltonian cycle in an unweighted graph
- Determining the minimal tour of cities – Travelling Salesperson problem
- Scheduling with profits and deadlines, to maximize the profit.
- Cliques in Social Networks - human groups form *cliques* on the basis of age, gender, race, ethnicity, religion/ideology, and many other things
- What is a clique in a graph G(V,E) ?

Problem k-Clique

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge

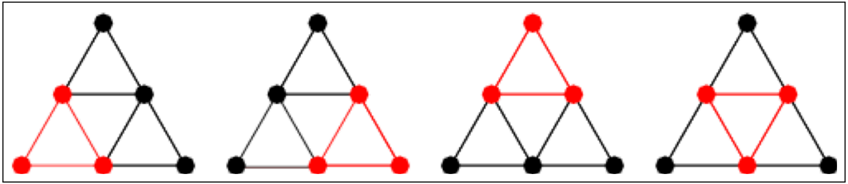


Figure: 3-Clique

Problem k-Clique

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE

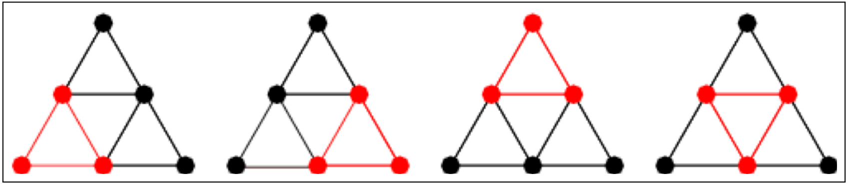


Figure: 3-Clique

Problem k-Clique

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE
  - Given a graph G=(V,E), if k-vertices in a graph are connected to each other then, it is a k–clique . . . .e.g.

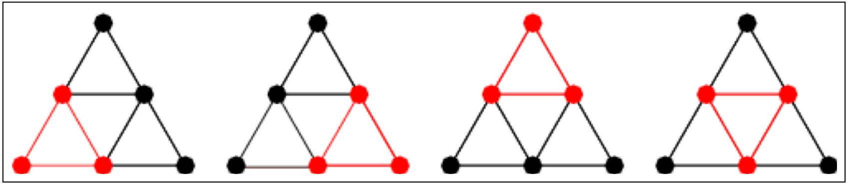


Figure: 3-Clique

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge

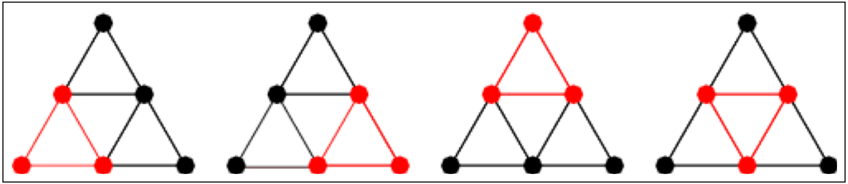


Figure: 3-Clique

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE

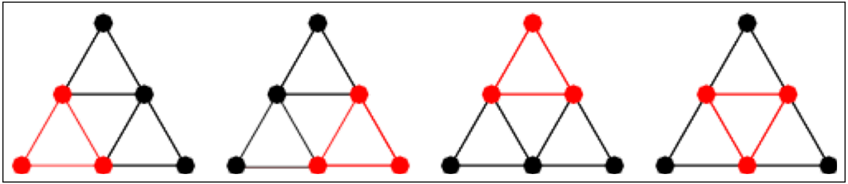


Figure: 3-Clique

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE
  - Given a graph G=(V,E), if k-vertices in a graph are connected to each other then, it is a k–clique . . . .e.g.

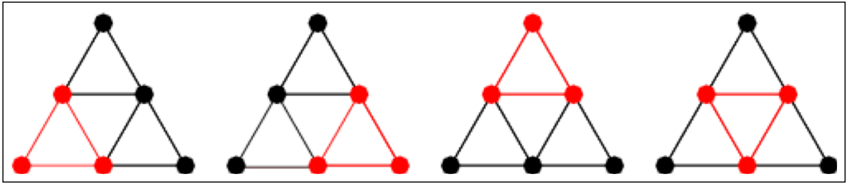


Figure: 3-Clique

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge

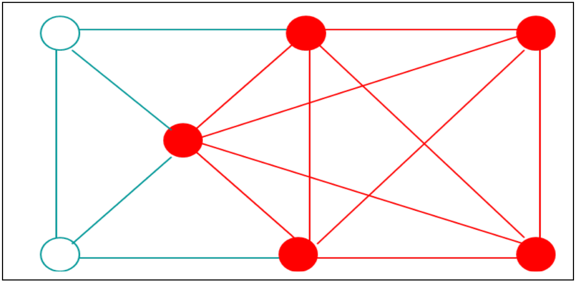


Figure: 5-Clique OR 6-clique?

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE

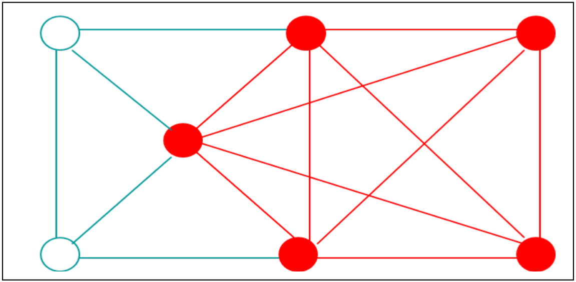


Figure: 5-Clique OR 6-clique?

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE
  - Is it 5-clique or 6-clique?

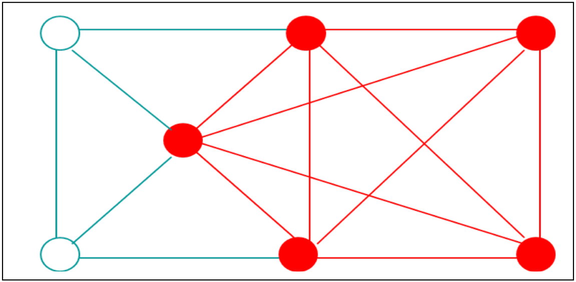


Figure: 5-Clique OR 6-clique?

Problem k-Clique...

- A graph G=(V, E) is a Clique if every two nodes in V are connected by an edge
- K-CLIQUE
  - Is it 5-clique or 6-clique?
  - What if the edges a-c, a-e, a-f are added ?

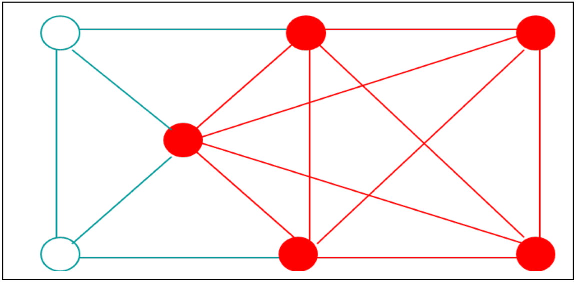


Figure: 5-Clique OR 6-clique?

Maximal and Maximum Clique

- Maximal clique in a graph

Maximal and Maximum Clique

- Maximal clique in a graph
  - is a clique that cannot be extended by including one more adjacent vertex,

Maximal and Maximum Clique

- Maximal clique in a graph
  - is a clique that cannot be extended by including one more adjacent vertex,
  - that is, a clique which does not exist exclusively within the vertex set of a larger clique.

Maximal and Maximum Clique

- Maximal clique in a graph
  - is a clique that cannot be extended by including one more adjacent vertex,
  - that is, a clique which does not exist exclusively within the vertex set of a larger clique.
- A maximum clique in a graph

Maximal and Maximum Clique

- Maximal clique in a graph
  - is a clique that cannot be extended by including one more adjacent vertex,
  - that is, a clique which does not exist exclusively within the vertex set of a larger clique.
- A maximum clique in a graph
  - is a clique of the largest possible size in a given graph.

Maximal and Maximum Clique...

- Maximum clique - triangle 1,2,5

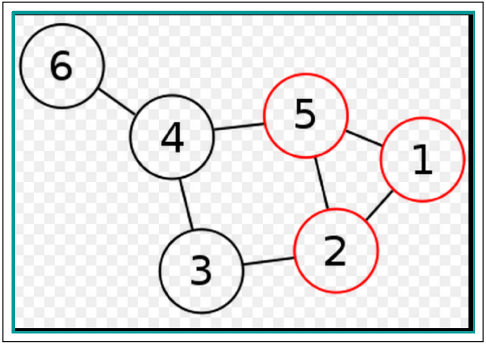


Figure: Maximum/Maximal clique

Maximal and Maximum Clique...

- Maximum clique - triangle 1,2,5
- Four Maximal cliques - pairs of (2,3), (3,4),(4,5), (4,6)

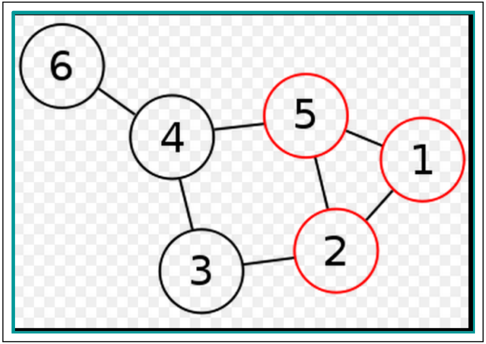


Figure: Maximum/Maximal clique

Maximal and Maximum Clique...

- Clique number $\omega(G)$ of a graph is the number of vertices in a maximum clique in G.

Maximal and Maximum Clique...

- Clique number $\omega(G)$ of a graph is the number of vertices in a maximum clique in G.
- Note the clique and independent set problems...... Are they related ?

The clique and independent set problems

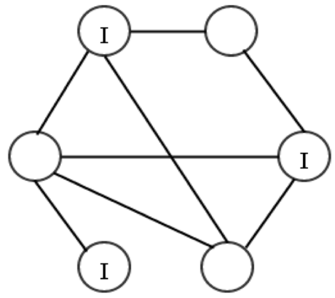
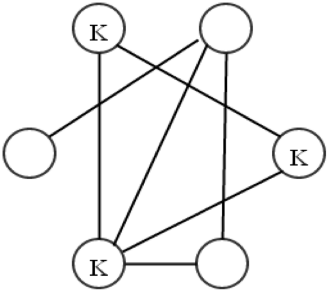


Figure: IS it a clique or an independent set ?

The clique and independent set problems



IS same vetices is forming k clicque in complement graph

Figure: IS it a clique or an independent set ?

Boolean Satisfiability : Terminologies

- Propositional (boolean) variable - a variable that may be assigned value true or false
- Literal - A Boolean variable or its negation.
- Propositional formula - an expression that is either a propositional variable or a propositional constant or an expression of boolean operator and its operands
- Clause : a disjunction – sequence of literals separated by V
- Conjunctive normal form - A regular form of propositional formula ϕ that is the conjunction of clauses

Boolean Satisfiability ...

- An example propositional CNF formula is
  $\phi = (\bar{x_1} \vee x_2 \vee x_3)\ (x_1 \vee \bar{x_2} \vee x_3)\ (\ x_2 \vee x_3)$
  where x_1, x_2, x_3 are propositional variables
- Truth assignment - a boolean valued function on the set i.e.
  an assignment of values true or false to each propositional variable in the set
- Satisfiability - When does a truth assignment is said to satisfy a formula ?
- SAT - Given a CNF formula ϕ, does it have a satisfying truth assignment?

Boolean Satisfiability Problem

- Input : A boolean formula F in CNF
- Goal:
  - Check if F is satisfiable or not.
  - e.g. if $F = (x_1 + x_2)$ can we assign at least one set of values to the literals of the formula so that $F = 1$.
- How to solve this problem deterministically ?
- What could be the Brute-force approach ?
- Time complexity ??
- $O(2^n|F|)$
- Cannot do any better than that.

Applications

- Bioinformatics

Applications

- Bioinformatics
  - clustering gene expression data

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction
- Electrical engineering

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction
- Electrical engineering
  - analyzing communications networks,

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction
- Electrical engineering
  - analyzing communications networks,
  - designing efficient circuits for computing partially specified Boolean functions.

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction
- Electrical engineering
  - analyzing communications networks,
  - designing efficient circuits for computing partially specified Boolean functions.
  - automatic test pattern generation

Applications

- Bioinformatics
  - clustering gene expression data
  - modelling ecological niches in food webs.
  - modelling protein structure prediction
- Electrical engineering
  - analyzing communications networks,
  - designing efficient circuits for computing partially specified Boolean functions.
  - automatic test pattern generation
  - finding a hierarchical partition of an electronic circuit into smaller

A few other hard problems

- Aerospace engineering

A few other hard problems

- Aerospace engineering
    - optimal mesh partitioning for finite elements.

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering
  - equilibrium of urban traffic flow.

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering
  - equilibrium of urban traffic flow.
- Economics

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering
  - equilibrium of urban traffic flow.
- Economics
  - computation of arbitrage in financial markets with friction.

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering
  - equilibrium of urban traffic flow.
- Economics
  - computation of arbitrage in financial markets with friction.
- Electrical engineering

A few other hard problems

- Aerospace engineering
  - optimal mesh partitioning for finite elements.
- Biology
  - protein folding.
- Chemical engineering
  - heat exchanger network synthesis.
- Civil engineering
  - equilibrium of urban traffic flow.
- Economics
  - computation of arbitrage in financial markets with friction.
- Electrical engineering
  - VLSI layout

A few other hard problems...

- Environmental engineering

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.

A few other hard problems...

- Environmental engineering
    - optimal placement of contaminant sensors.
- Financial engineering
    - find minimum risk portfolio of given return.
- Game theory
    - find Nash equilibrium that maximizes social welfare.
- Genomics

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.
- Genomics
  - phylogeny reconstruction.

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.
- Genomics
  - phylogeny reconstruction.
- Mechanical engineering

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.
- Genomics
  - phylogeny reconstruction.
- Mechanical engineering
  - structure of turbulence in sheared flows.

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.
- Genomics
  - phylogeny reconstruction.
- Mechanical engineering
  - structure of turbulence in sheared flows.
- Medicine

A few other hard problems...

- Environmental engineering
  - optimal placement of contaminant sensors.
- Financial engineering
  - find minimum risk portfolio of given return.
- Game theory
  - find Nash equilibrium that maximizes social welfare.
- Genomics
  - phylogeny reconstruction.
- Mechanical engineering
  - structure of turbulence in sheared flows.
- Medicine
  - reconstructing 3-D shape from biplane angiocardiogram.

A few other hard problems...

- Operations research

A few other hard problems...

- Operations research
  - optimal resource allocation.

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.

A few other hard problems...

- Operations research
    - optimal resource allocation.
- Physics
    - partition function of 3-D Ising model in statistical mechanics.
- Politics

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.
- Politics
  - Shapley-Shubik voting power.

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.
- Politics
  - Shapley-Shubik voting power.
- Pop culture

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.
- Politics
  - Shapley-Shubik voting power.
- Pop culture
  - Minesweeper consistency.

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.
- Politics
  - Shapley-Shubik voting power.
- Pop culture
  - Minesweeper consistency.
- Statistics

A few other hard problems...

- Operations research
  - optimal resource allocation.
- Physics
  - partition function of 3-D Ising model in statistical mechanics.
- Politics
  - Shapley-Shubik voting power.
- Pop culture
  - Minesweeper consistency.
- Statistics
  - optimal experimental design.

content...

content...

content...

content...