Greedy Algorithms

GeeksforGeeks
A computer science portal for geeks

amount = 57

coins[] = {5, 10, 2, 1}

① After Sorting:

coins[] = {10, 5, 2, 1}

② res = 0

③ i = 0 : res = 5, amount = 7

i = 1 : res = 6, amount = 2

i = 2 : res = 7, amount = 0

```
int minCoins(int coins[], int n, int amount)
{
  ① Sort coins[] in decreasing order.

  ② int res = 0;

  ③ for(int i = 0; i < n; i++)
    {
      if (coin[i] ≤ amount)
      {
        int c = floor(amount/coin[i]);
        res = res + c;
        amount = amount - c * coin[i];
      }
      if (amount == 0)
        break;
    }

  ④ Return res;
}
```

Greedy Algorithms

GeeksforGeeks
A computer science portal for geeks

General Structure

```
getOptimal(Item arr[], int n)

① Initialize : res = 0
② While (All items are not considered)
   {
       i = selectAnItem()
       if (feasible(i))
           res = res + i
   }
③ Return res
```

Greedy Algorithms

GeeksforGeeks
A computer science portal for geeks

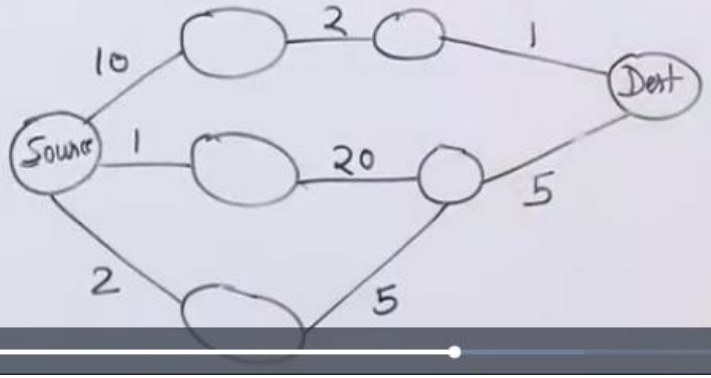
Greedy algorithms may not work always

Consider

coins[] = {18, 1, 10}

amount = 20

Another Example Problem : Longest Path



Greedy Algorithms

GeeksforGeeks
A computer science portal for geeks

Applications:

- Finding Optimal Solutions
  - Activity Selection
  - Fractional Knapsack
  - Job Sequencing
  - Prim's Algorithm
  - Kruskal's Algorithm
  - Dijkstra's Algorithm
  - Huffman Coding
- Finding Close to Optimal Solutions for NP Hard Problems like Travelling Salesman Problem.

Activity Selection Problem

GeeksforGeeks
A computer science portal for geeks

I/p : {(2, 3), (1, 4), (5, 8), (6, 10)}

O/p : 2

I/p : {(1, 3), (2, 4), (3, 8), (10, 11)}

O/p : 3

Maximum no. of activities that can happen on a single tasking machine.

Activity Selection Problem

GeeksforGeeks
A computer science portal for geeks

I/p: {(3, 8), (2, 4), (1, 3), (10, 11)}

① Sort according to finish time:

{(1, 3), (2, 4), (3, 8), (10, 11)}

② Initialize Solution as first activity

③ Do following for remaing activities

(a) If current activity overlaps with the last picked activity in the solution, ignore the current activity

(b) Else add the current activity to the solution.

Activity Selection Problem

GeeksforGeeks
A computer science portal for geeks

I/p: {(3, 8), (2, 4), (1, 3), (10, 11)}

① Sort according to finish time:

{(1, 3), (2, 4), (3, 8), (10, 11)}

② Initialize Solution as first activity

③ Do following for remaing activities

(a) If current activity overlaps with the last picked activity in the solution, ignore the current activity

(b) Else add the current activity to the solution.

$S = \{(1, 3)\}$

$S = \{(1, 3), (3, 8)\}$

$S = \{(1, 3), (3, 8), (10, 11)\}$

Activity Selection Problem

GeeksforGeeks
A computer science portal for geeks

I/p: {(3, 8), (2, 4), (1, 3), (10, 11)}

$S1 = \{(x_1, x_2), (x_3, x_4) \ldots\}$ — $n1$

$S2 = \{(y_1, y_2), (y_3, y_4) \ldots\}$ — $n2 - 1$

$n2 > n1$

① Sort according to finish time:

② Initialize Solution as first activity

③ Do following for remaining

(a) If current activity ... on,
the last picked acti...
ignore the current activi...

(b) Else add t...
solution.

always get the

Activity Selection Problem

GeeksforGeeks
A computer science portal for geeks

I/p: {(3, 8), (2, 4), (1, 3), (10, 11)}

1. Sort according to finish time:

2. Initialize Solution as first activity

3. Do following for remaining activities
   (a) If current activity overlaps with the last picked activity in the solution, ignore the current activity
   (b) Else add the current activity to the solution.

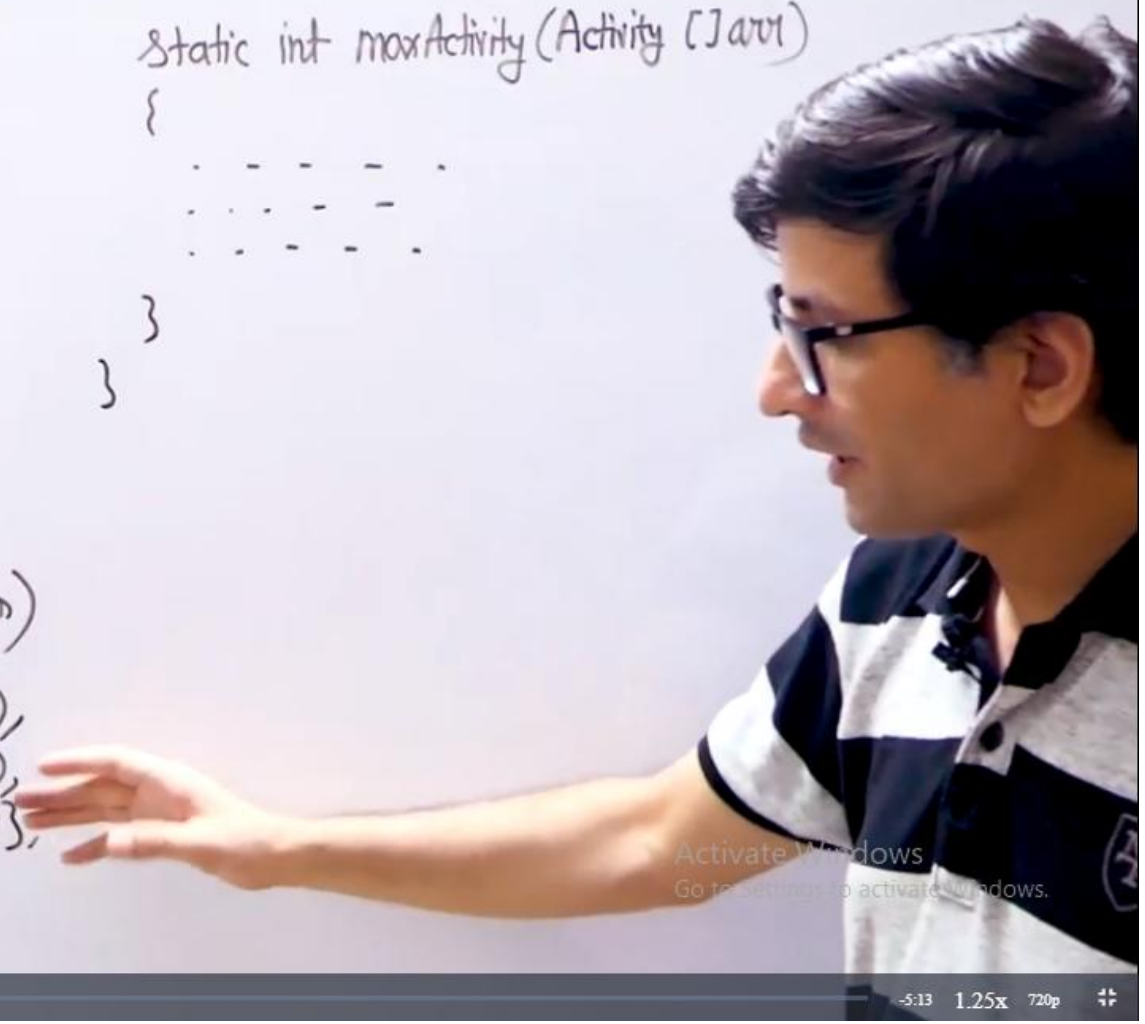
Activity Selection Problem (Java)

GeeksforGeeks
A computer science portal for geeks

```
class Activity
{
  int start;
  int finish;
  Activity(int s, int f)
  {
    start = s;
    finish = f;
  }
}
class Test
{
    public static void main(String [Jargs)
    {
        Activity arr[] = {new Activity(12, 25),
                          new Activity(10, 20),
                          new Activity(20, 30)};

        System.out.println(maxActivity(arr));
    }
```

```
static int maxActivity(Activity [Jarr)
{
    . - - - - .
    . . . - -
    . - - - - .
}
```



Activity Selection Problem (Java)

GeeksforGeeks
A computer science portal for geeks

```
class Activity
{
  int start;
  int finish;
  Activity(int s, int f)
  {
    start = s;
    finish = f;
  }
}
```

```
class MyCmp implements Comparator<Activity>
{
  public int compare(Activity a1, Activity a2)
  {
    return a1.finish - a2.finish;
  }
}
```

{(12, 25), (10, 20), (20, 30)}

After Sorting:

(10, 20) (12, 25) (20, 30)

prev = 0, res = 1

curr = 1: Ignore

curr = 2: res = 2, prev = 2

```
static int maxActivity(Activity [] arr)
{
  Arrays.sort(arr, MyCmp);
  int res = 1;
  int prev = 0;
  for (int curr = 1; curr < arr.length; curr++)
  {
    if (arr[curr].start >= arr[prev].finish)
    {
      res++;
      prev = curr;
    }
  }
  return res;
}
```

001 What is Greedy Algorithm

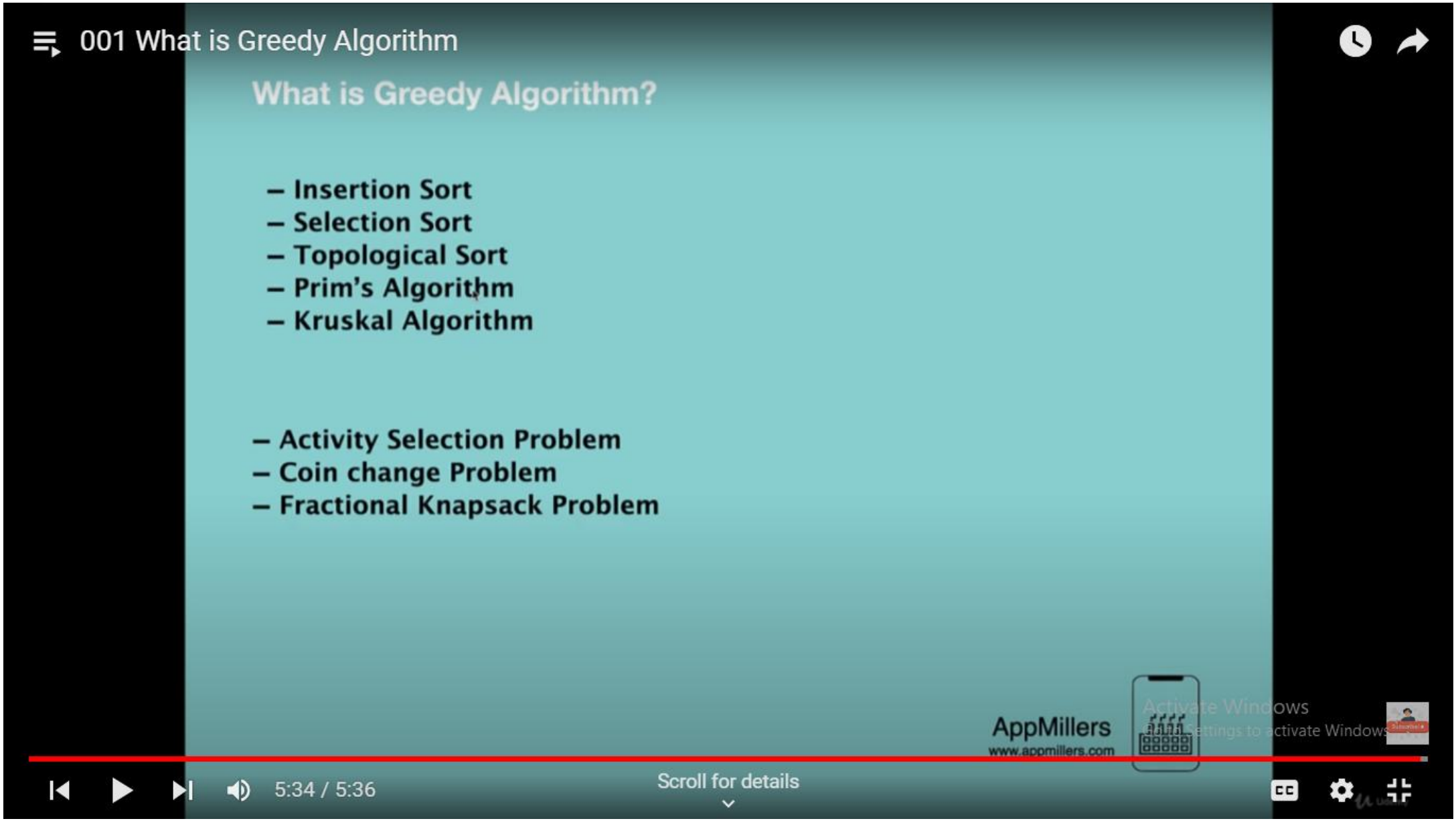
What is Greedy Algorithm?

- It is an algorithmic paradigm that builds the solution piece by piece
- In each step it chooses the piece that offers most obvious and immediate benefit
- It fits perfectly for those solutions in which local optimal solutions lead to global solution

AppMillers
www.appmillers.com

4:50 / 5:36

Scroll for details


001 What is Greedy Algorithm
What is Greedy Algorithm?
– Insertion Sort
– Selection Sort
– Topological Sort
– Prim's Algorithm
– Kruskal Algorithm
– Activity Selection Problem
– Coin change Problem
– Fractional Knapsack Problem
AppMillers
www.appmillers.com
Activate Windows
Go to Settings to activate Windows.
5:34 / 5:36
Scroll for details

002 Known Greedy Algorithms

Greedy Algorithms

- Insertion Sort
- Selection Sort
- Topological Sort
- Prim's Algorithm
- Kruskal Algorithm

002 Known Greedy Algorithms

Greedy Algorithms

Insertion Sort

AppMillers
www.appmillers.com

1:56 / 8:30

Scroll for details

002 Known Greedy Algorithms

Greedy Algorithms

Selection Sort

AppMillers
www.appmillers.com

2:55 / 8:30

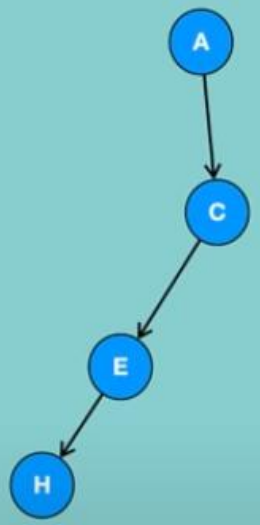
Scroll for details

002 Known Greedy Algorithms

Greedy Algorithms

Topological Sort

A C E H



Stack

AppMillers
www.appmillers.com

4:45 / 8:30

Scroll for details

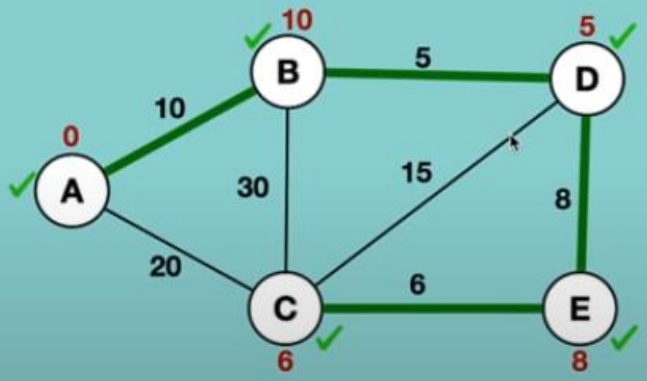
002 Known Greedy Algorithms

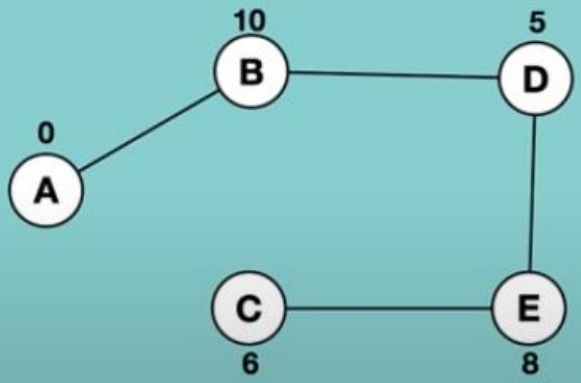
Greedy Algorithms

Prims Algorithm

It is a greedy algorithm
It finds a minimum spanning tree for weighted undirected graphs in following ways

1. Take any vertex as a source set its weight to 0 and all other vertices' weight to infinity
2. For every adjacent vertices if the current weight is more than current edge then we set it to current edge
3. Then we mark current vertex as visited
4. Repeat these steps for all vertices in increasing order of weight





AppMillers
www.appmillers.com

6:30 / 8:30

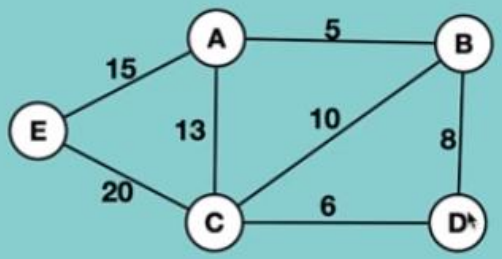
Scroll for details

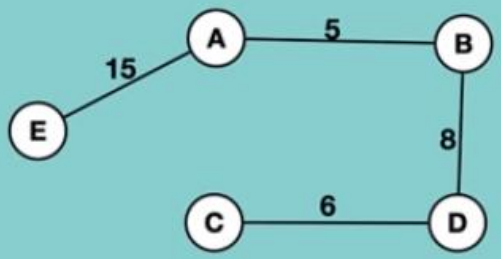
Greedy Algorithms

Kruskal's Algorithm

It is a greedy algorithm
It finds a minimum spanning tree for weighted undirected graphs in two ways

- Add increasing cost edges at each step
- Avoid any cycle at each step





AppMillers
www.appmillers.com

003 Activity Selection Problem

Activity selection problem

Given N number of activities with their start and end times. We need to select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Activity	A1	A2	A3	A4	A5	A6
Start	0	3	1	5	5	8
Finish	6	4	2	8	7	9

2 Activities

Activity	A3	A2	A1	A5	A4	A6
Start	1	3	0	5	5	8
Finish	2	4	6	7	8	9

4 Activities

AppMillers
www.appmillers.com

5:00 / 5:36

Scroll for details

Activity selection problem

```
Sort activities based on finish time
```

```
Select first activity from sorted array and print it
```

```
For all remainin activities:
   If the start time of this activity is greater or equal to the
   finish time of previously selected activity then select this
   activity and print it
```

AppMillers
www.appmillers.com

005 Coin Change Problem

Coin Change Problem

You are given coins of different denominations and total amount of money. Find the minimum number of coins that you need yo make up the given amount.

Infinite supply of denominations : {1,2,5,10,20,50,100,1000}

Total amount : 2035

2035 - 1000 = 1035

1035 - 1000 = 35

35 - 20 = 15

15 - 10 = 5

5 - 5 = 0

Result : 1000, 1000, 20, 10, 5

Answer: 5

3:51 / 4:26

Scroll for details

Coin Change Problem

Find the biggest coin that is less than given total number

Add coin to the result and subtract coin from total number

```
If V is equal to zero:
     Then print result
else:
    Repeat Step 2 and 3
```

AppMillers
www.appmillers.com

007 Fractional Knapsack Problem

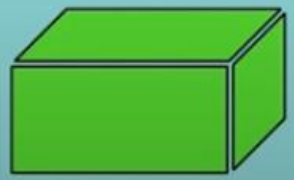
Fractional Knapsack Problem

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

30 kg

10 kg, Value : 20

20 kg, Value : 10



10 kg, Value : 30

AppMillers
www.appmillers.com

1:26 / 5:22

Scroll for details

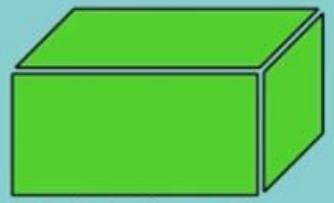
007 Fractional Knapsack Problem

Fractional Knapsack Problem

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

20 kg, Value : 100

30 kg, Value : 120



10 kg, Value : 60

50 kg

100 + 120 = 220

60 + 100 + 120*2/3 = 240

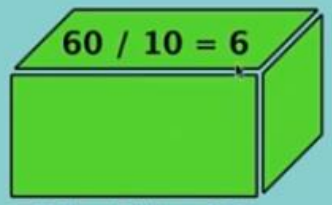
AppMillers
www.appmillers.com

3:20 / 5:22

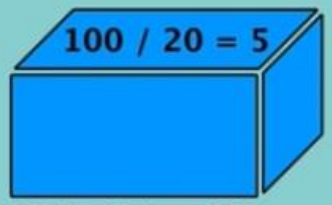
Scroll for details

Fractional Knapsack Problem

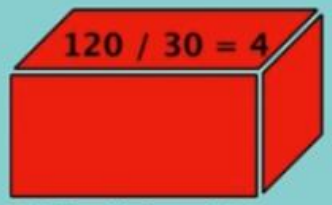
Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.



10 kg, Value : 60



20 kg, Value : 100



30 kg, Value : 120

50 kg

100 + 120 = 220

60 + 100 + 120*2/3 = 240

AppMillers
www.appmillers.com

007 Fractional Knapsack Problem

Fractional Knapsack Problem

- Calculate the density or ratio for each item
- Sort items based on this ratio
- Take items with the highest ratio sequentially until weight allows
- Add the next item as much (fractional) as we can

AppMillers
www.appmillers.com

Scroll for details