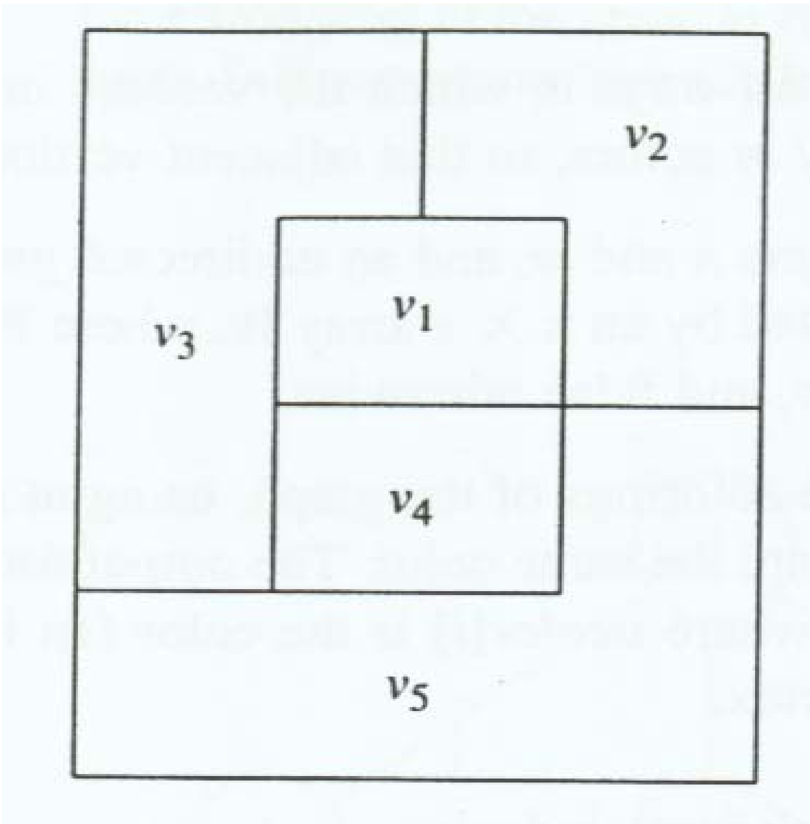
Backtracking (2)

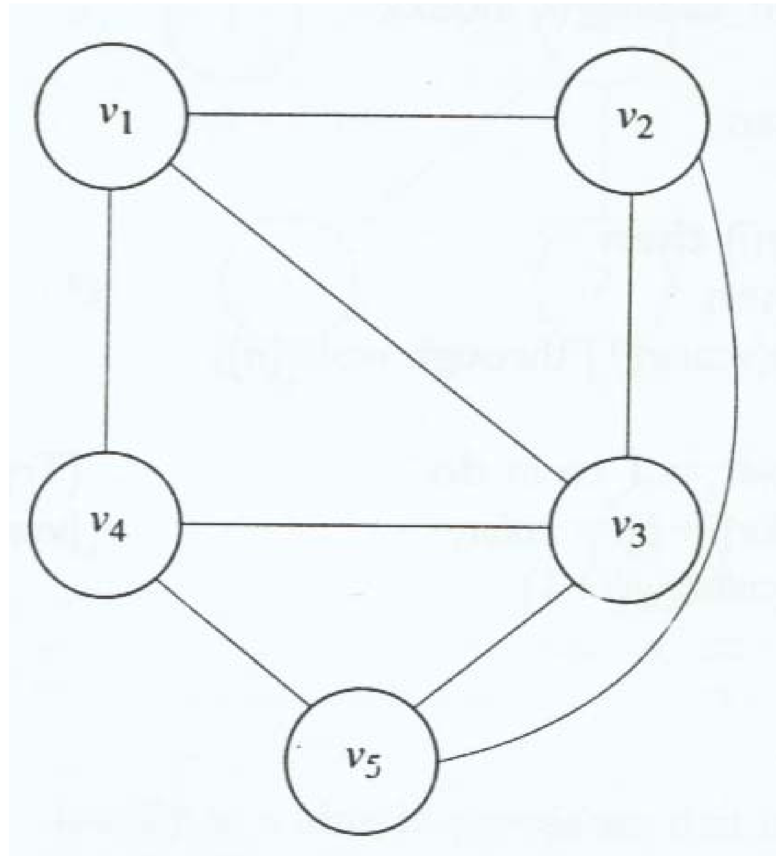
Graph Coloring Problem

Goal :

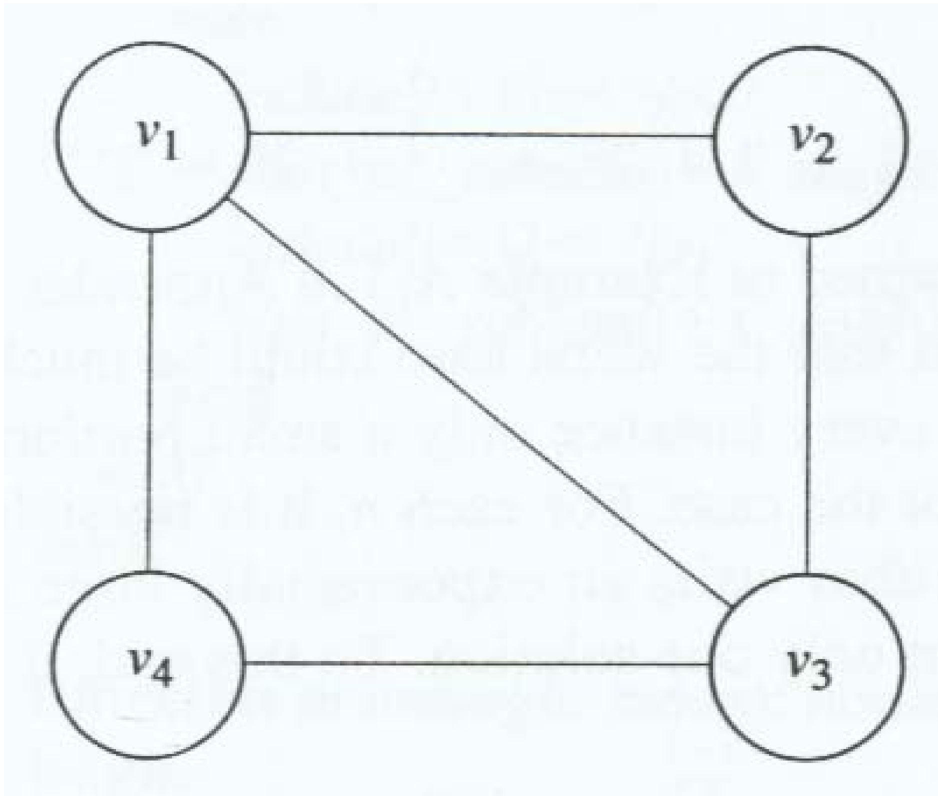
Finding all the ways to color an undirected graph using at most m defferent colors, so that no two adjacent vertices are the same color.

Application: coloring the map

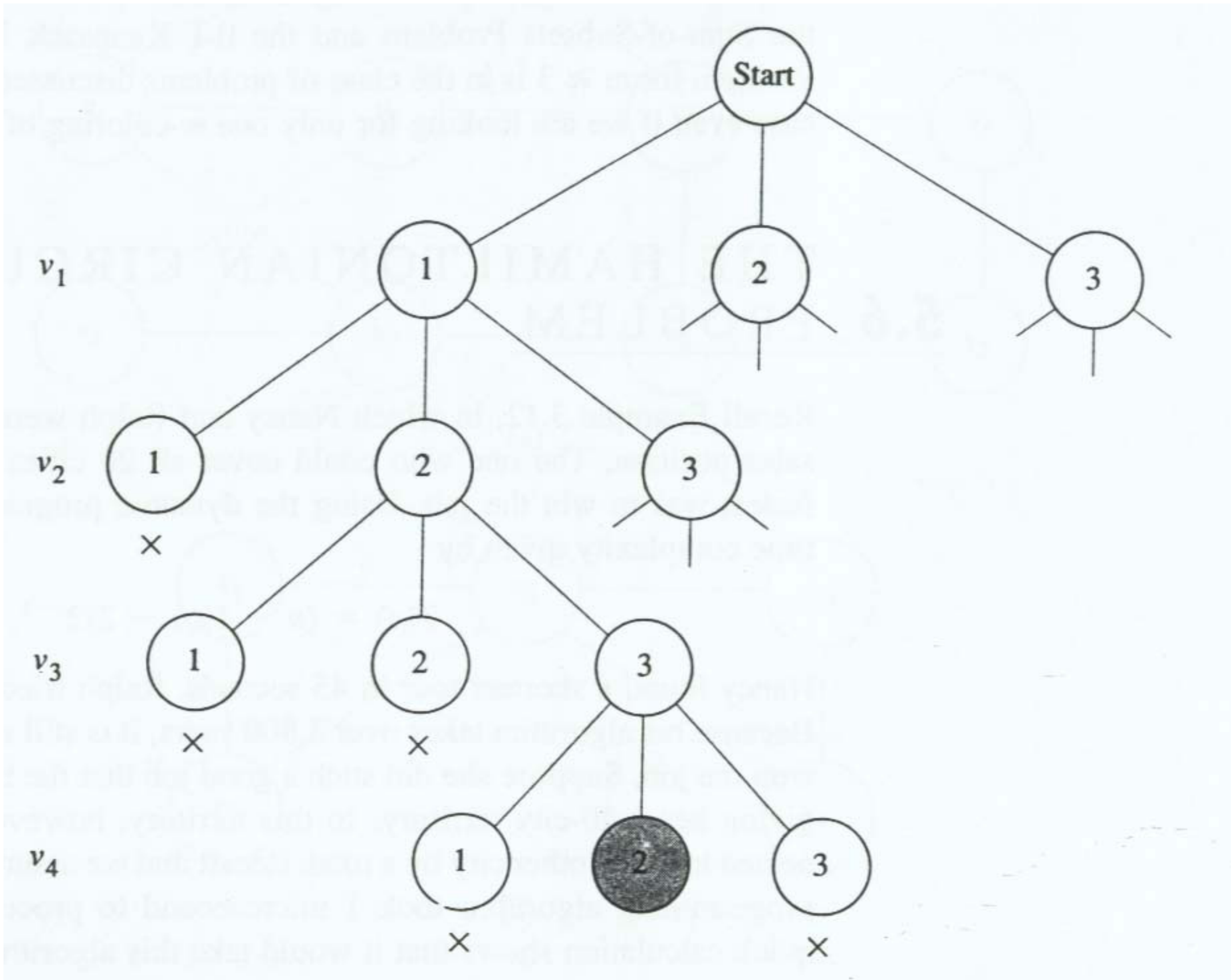




3-Coloring Problem



3-Coloring Problem



The *m*-Coloring Graph Algorithm

```
Procedure m_coloring(i:index);
Var color: integer
Begin
  if promising(i) then
      if i=n then
            write(vcolor[i] through vcolor[n])
      else
            for color:= 1 to m do
                  vcolor[i+1]:=color;
                  m_coloring(i+1)
            end
      end
  end
End;
```

The *m*-Coloring Graph Algorithm

```
function promising(i:index):boolean;
Var j:index;
Begin
   j:=1;
   promising:=true;
   while j<i and promising do
       if W[i,j] and vcolor[i]=vcolor[j] then
            promising:=false
       end
       j:=j+1
   end
End;
```

0-1 Knapsack

- Objective : determine a set of items that maximizes the total profit under the constraint that the total weight cannot exceed W
- First, order the items in non decreasing order according *pi/wi*

Properties of Knapsack's State Space Tree

- profit : sum of profit of items up to the node
- weight : sum of weight of those items
- maxprofit
- bound
- total weight

- A node is **non-promising if *weight >=W and bound ≤ maxprofit***
- Steps :
  - Initializes bound ← profit and totweight ←weight
  - When grab items, add the profit to bound and weight to totweight, until get item that bring totweight above W
  - Get the fraction of that item, add to bound

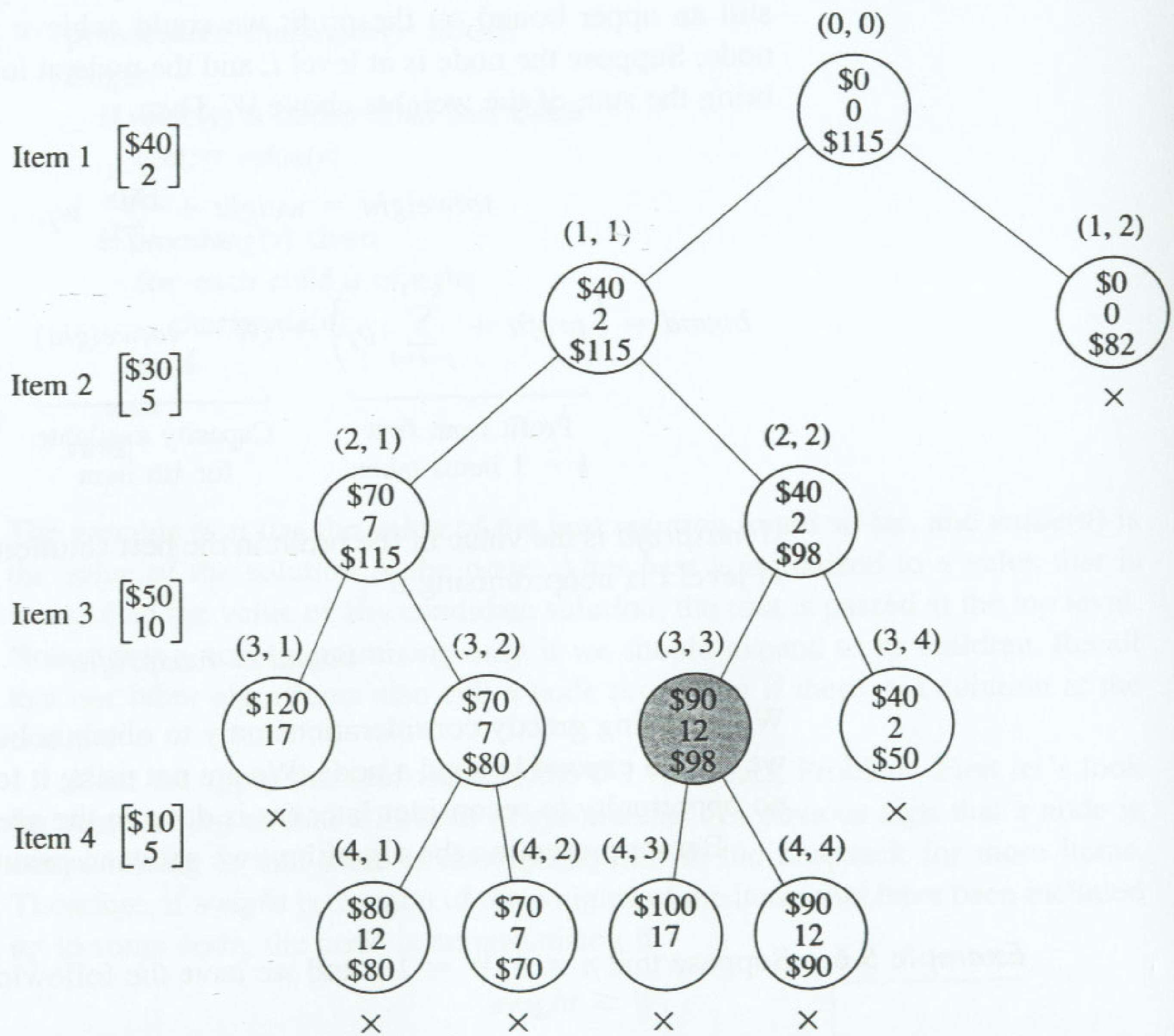
Suppose the node at level I, and the node at level k is
the one that would bring the sum of the weight above W, then :

$$totweight = weight + \sum_{j=i+1}^{k-1} w_j, \qquad \text{and}$$

$$bound = \underbrace{\left(profit + \sum_{j=i+1}^{k-1} p_j\right)}_{\substack{\text{Profit from first} \\ k-1 \text{ items taken}}} + \underbrace{(W - totweight)}_{\substack{\text{Capacity available} \\ \text{for } k\text{th item}}} \times \underbrace{\frac{p_k}{w_k}}_{\substack{\text{Profit per unit} \\ \text{weight for } k\text{th item}}}.$$

0/1 Knapsack : Case-1

i	Pi	wi	Pi/wi
1	$40	2	$20
2	$30	5	$6
3	$50	10	$5
4	$10	5	$2



Item 1 [$40 2]
Item 2 [$30 5]
Item 3 [$50 10]
Item 4 [$10 5]
(0, 0)
$0
0
$115
(1, 1)
$40
2
$115
(1, 2)
$0
0
$82
(2, 1)
$70
7
$115
(2, 2)
$40
2
$98
(3, 1)
$120
17
(3, 2)
$70
7
$80
(3, 3)
$90
12
$98
(3, 4)
$40
2
$50
(4, 1)
$80
12
$80
(4, 2)
$70
7
$70
(4, 3)
$100
17
(4, 4)
$90
12
$90

1. Maxprofit = \$0
2. Visit (0,0)

   a. profit = \$0, weight = 0

   b. bound, i=0, and k=3 (because 2+5+7 = 17>16)

$$totweight = weight + \sum_{j=0+1}^{3-1} wj = 0 + 2 + 5 = 7$$

$$bound = profit + \sum_{j=0+1}^{3-1} Pj + (w - totweight)x(P3/w3)$$

$$= \$0 + \$40 + \$30 + (16 - 7)x\$50/10 = \$115$$

   c. Promising ?

      weight < W and bound > maxprofit ➔promising

3. Visit (1,1)

a. profit = \$0 + \$40 = \$40, and weight = 0 + 2 = 2

b. maxprofit = \$40

c. bound, i=1, and k=3 (because 2+5+7 = 17>16)

$$totweight = weight + \sum_{j=1+1}^{3-1} wj = 2+5=7$$

$$bound = profit + \sum_{j=1+1}^{3-1} Pj + (w - totweight)x(P3/w3)$$

$$= \$40 + \$30 + (16 - 7)x\$50/10 = \$115$$

c. Promising ?

weight < W and bound > maxprofit ➔promising

4. Visit (2,1)

   a. profit = \$40+\$30 = \$70, and weight = 2+5 = 7

   b. maxprofit = \$70

   c. bound, i=2, and k=3 (because 7+10 = 17>16)

$$totweight = weight + \sum_{j=2+1}^{3-1} wj = 7 + 0 = 7$$

$$bound = profit + \sum_{j=2+1}^{3-1} Pj + (w - totweight)x(P3 / w3)$$

$$= \$70 + (16 - 7)x\$50 / 10 = \$115$$

   c. Promising ?

      weight < W and bound > maxprofit ➔promising

5. Visit (3,1)

   a. profit = \$70+\$50 = \$120, and weight = 7+10=17

   b. weight > W, so maxprofit doesn't change and bound not computed

   c. Promising ?

   weight > W ➔ nonpromising

6. Back to node (2,1)

7. Visit (3,2)

a. profit = \$70, and weight =7

b. maxprofit = \$70

c. bound, i=3, and k=5

$$totweight = weight + \sum_{j=3+1}^{5-1} wj = 7 + 5 = 12$$

$$bound = profit + \sum_{j=3+1}^{5-1} Pj = \$70 + \$10 = \$80$$

c. Promising ?

weight < W and bound > maxprofit ➔promising

8. Visit (4,1)

a. profit = \$80, and weight =12

b. maxprofit = \$80

c. bound, i=4, and k=5

$$totweight = weight + \sum_{j=4+1}^{5-1} wj = 12 + 0 = 12$$

$$bound = profit + \sum_{j=4+1}^{5-1} Pj = \$80 + 0 = \$80$$

c. Promising ?

weight < W and bound = maxprofit ➔nonpromising

9. Visit (4,1)
    a. profit = \$70, and weight =7
    b. maxprofit = \$80
    c. bound, i=4, and k=5

$$totweight = weight + \sum_{j=4+1}^{5-1} wj = 7 + 0 = 7$$

$$bound = profit + \sum_{j=4+1}^{5-1} Pj = \$70 + 0 = \$70$$

c. Promising ?

weight < W and bound < maxprofit ➔nonpromising

10. Visit (2,2)

a. profit = \$40, and weight =2

b. maxprofit = \$80

c. bound, i=2, and k=4

$$totweight = weight + \sum_{j=2+1}^{4-1} wj = 2 + 10 = 12$$

$$bound = profit + \sum_{j=2+1}^{4-1} Pj + (W - totweight) x P4 / W4$$

$$= \$40 + \$50 + (16 - 12).\$10 / 5 = \$98$$

c. Promising ?

weight < W and bound > maxprofit ➔promising

11. Visit (3,3)

a. profit = \$90, and weight =12

b. maxprofit = \$90

c. bound, i=3, and k=4

$$totweight = weight + \sum_{j=3+1}^{4-1} wj = 12 + 0 = 12$$

$$bound = profit + \sum_{j=3+1}^{4-1} Pj + (W - totweight) x P4 / W4$$

$$= \$90 + \$0 + (16 - 12).\$10 / 5 = \$98$$

c. Promising ?

weight < W and bound > maxprofit ➔promising

12. Visit (4,3)
    a. profit = \$100, and weight =17
    b. nonpromising, maxprofit = \$90
13. Visit (4,2)
    a. profit = \$90, and weight =12
    b. bound = \$90 ➔ nonpromising
14. Visit(3,4)
    a. profit = \$40, and weight =2
    b. bound = \$50 ➔ nonpromising
15. Visit (1,2)
    a. profit = \$0, and weight =0
    b. bound = \$82 ➔ nonpromising

Algoritma

```
procedure knapsack (i: index;
          profit, weight: integer);

begin
  if weight ≤ W and profit > maxprofit then        {This set is best so far.}
    maxprofit:= profit;
    numbest:=i;                                    {Set numbest to number}
    bestset:= include                              {of items considered. Set}
  end;                                             {bestset to this solution.}
  if promising(i) then
    include[i+1]:= 'yes';                          {Include w[i + 1].}
    knapsack(i+1, profit+p[i+1], weight+w[i+1]);
    include[i+1]:= 'no';                           {Do not include w[i + 1].}
    knapsack(i+1, profit, weight)
  end
end;
```

Algoritma

```
function promising(i: index): boolean;
var
  j,k: index;
  totweight: integer;
  bound: real;
begin
  if weight ≥ W then                          {Node is promising only}
    promising:= false                         {if we should expand to}
  else                                        {its children. There must}
    j:=i+1;                                   {be some capacity left for}
    bound:= profit;                           {the children.}
totweight:= weight;
    while j ≤ n and totweight + w[j] ≤ W do   {Grab as many items as}
      totweight:= totweight + w[j];           {possible.}
      bound:= bound + p[j];
      j:= j+1
    end;
    k:= j;                                    {Use k for consistency}
    if k ≤ n then                             {with formula in text.}
      bound:= bound + (W − totweight)*p[k]/w[k]   {Grab fraction of kth}
    end;                                      {item.}
    promising:= bound > maxprofit
  end
end;
```

```
numbest:= 0;
maxprofit:= 0;
knapsack(0,0,0);
write(maxprofit);                {Write the maximum profit.}
for i:=1 to numbest do
   write(bestset[i])             {Show which items are included}
end;                             {in an optimal set.}
```