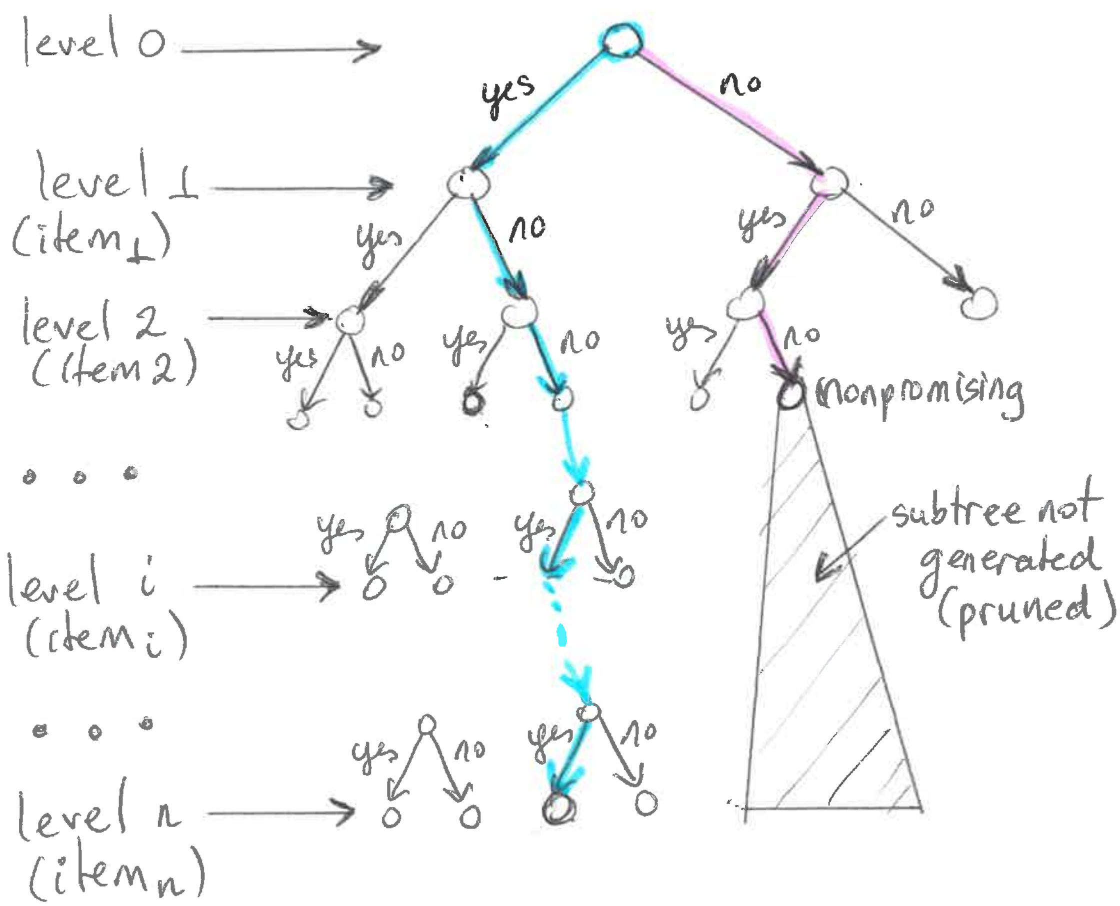
Solving the Knapsack problem with Branch-and-Bound

- formulate the solution as:

$$\text{solution} = (\underbrace{comp_1}_{\substack{\text{select item}_1 \\ (\text{yes/no})}}, \underbrace{comp_2}_{\substack{\text{select item}_2 \\ (\text{yes/no})}}, \ldots, \underbrace{comp_i}_{\substack{\text{select item}_i \\ (\text{yes/no})}}, \ldots, \underbrace{comp_n}_{\substack{\text{select item}_n \\ (\text{yes/no})}})$$

- design the search tree based on these components:



Knapsack problem

example

$n=4$

$W=16$

	1	2	3	4
w	2	5	10	5
p	40	30	50	10

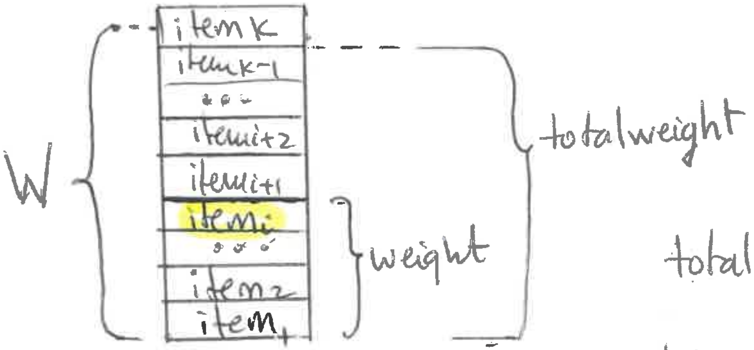
item	profit p_i	weight w_i	p_i/w_i
1	\$40	2	\$20
2	\$30	5	\$6
3	\$50	10	\$5
4	\$10	5	\$2

- assume that the items are sorted in decreasing order of p_i/w_i

Branch-and-Bound Search Tree

- for each node in the search tree
  - weight – total weight of the items selected so far
  - profit – total profit of the items selected so far
  - bound – upperbound on the profit that can be obtained with this partial solution
  - maxprofit – best profit so far
- if bound < maxprofit, then the node is nonpromising
- How do we compute the bound?

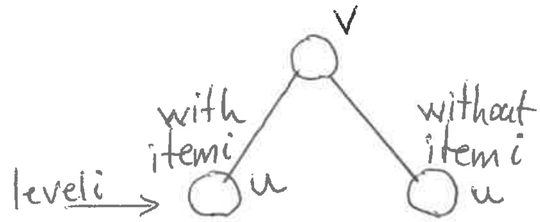
Assume that
- current node is at level i $(item_1, item_2, \ldots, item_i \ldots)$
- $item_k$ would bring the weight above W



$$totalweight = weight + \sum_{j=i+1}^{k-1} w_j$$

$$bound = profit + \sum_{j=i+1}^{k-1} p_j + (W - totalweight) \cdot \frac{p_k}{w_k}$$

Knapsack-BreadthFS-Branch-and-Bound(n, p[ ], w[ ], W, maxprofit)




```
Q = ∅
r.level = 0; r.profit = 0; r.weight = 0
maxprofit = 0
ENQUEUE(Q,r)
while Q ≠ ∅
        v = DEQUEUE(Q)
        u.level = v.level +1
        u.weight = v.weight + w[u.level]
        u.profit = v.profit + p[u.level]
        if (u.weight ≤ W and u.profit > maxprofit)
                maxprofit = u.profit
        if bound(u) > maxprofit
                ENQUEUE(Q,u)
        u.weight = v.weight
        u.profit = v.profit
        if bound(u) > maxprofit
                ENQUEUE(Q,u)
```

left child

set u as the child of v that includes the next item

right child

set u as the child of v that does not include the next item

RT analysis

- the number of nodes is upperbounded as follows:

$$\le 1 + 2 + 2^2 + 2^3 + \dots + 2^n = \frac{2^{n+1}-1}{2-1} = 2 \cdot 2^n - 1$$

geometric series

total number of nodes $= O(2^n)$

- bound() takes $O(n)$

$$\boxed{\text{total RT} = O(n \cdot 2^n)}$$

Branch-and-bound with Breadth FS

Knapsack problem

maxprofit = ~~0~~ ~~40~~ ~~70~~ 90

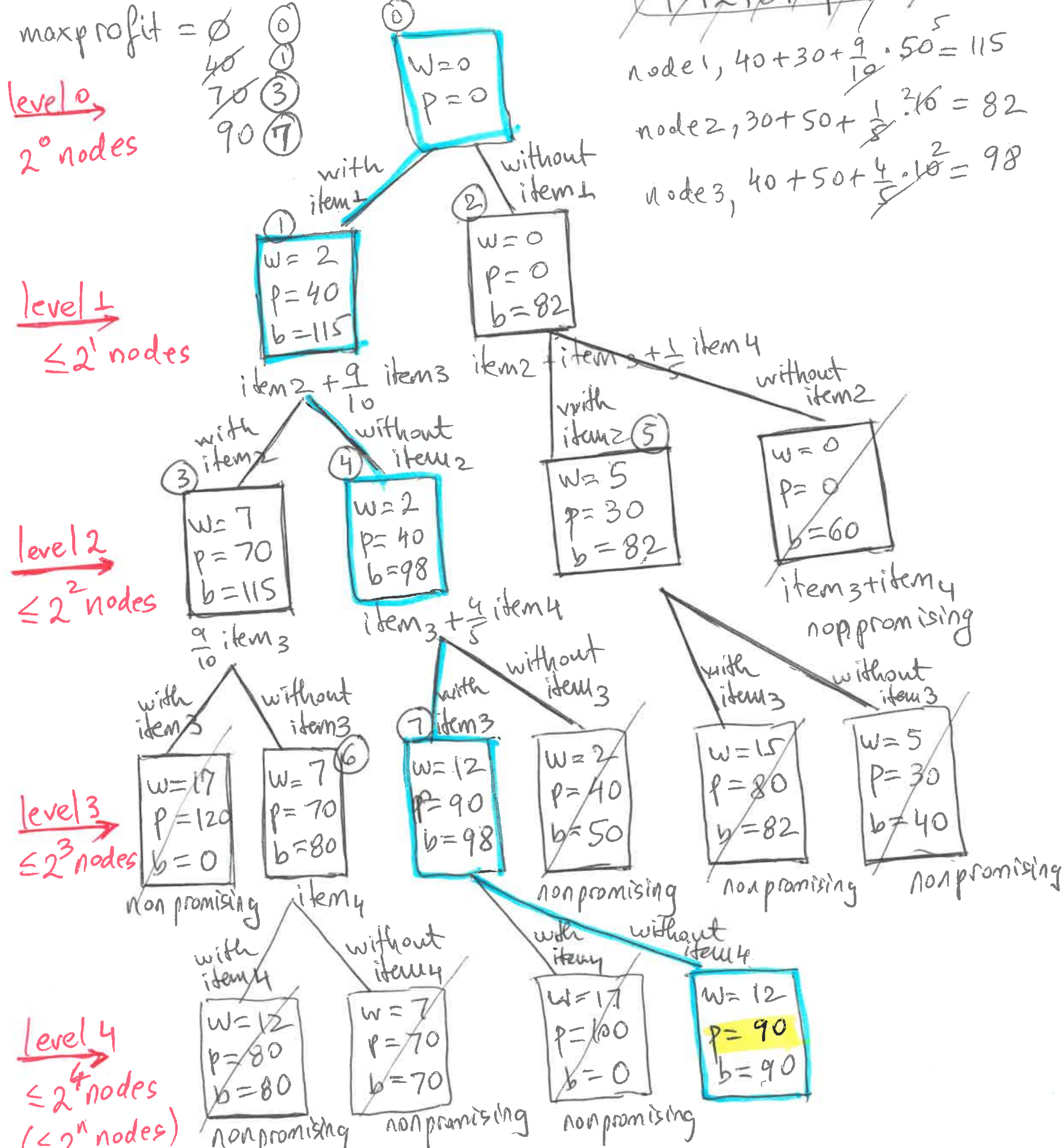
0 1 3 7

Q (FIFO): ~~0~~ | ~~1~~ | ~~2~~ | ~~3~~ | ~~4~~ | ~~5~~ | ~~6~~ | ~~7~~

node 1, $40 + 30 + \frac{9}{10} \cdot 50 = 115$

node 2, $30 + 50 + \frac{1}{5} \cdot 10 = 82$

node 3, $40 + 50 + \frac{4}{5} \cdot 10 = 98$



solution = (item1, item3), with weight = 12
profit = 90

Knapsack-BestFS-Branch-and-Bound(n,p[ ],w[ ],W,maxprofit)

```
PQ = ∅
r.level = r.profit = r.weight = 0
maxprofit = 0
r.bound = bound(r)
insert(PQ,r)
while PQ ≠ ∅
O(lg n) →    v = remove(PQ)
        if v.bound > maxprofit
            u.level = v.level +1
            u.weight = v.weight + w[u.level]
            u.profit = v.profit + p[u.level]      set u as the child of v that includes the next item
            if (u.weight ≤ W and u.profit > maxprofit)
                maxprofit = u.profit
O(n) →      u.bound = bound(u)
O(n) →      if bound(u) > maxprofit
O(lg n) →       insert(PQ,u)
            u.weight = v.weight
            u.profit = v.profit
O(n) →      u.bound = bound(u)                    set u as the child of v that does not include the next item
            if u.bound > maxprofit
O(lg n) →       insert(PQ,u)
```

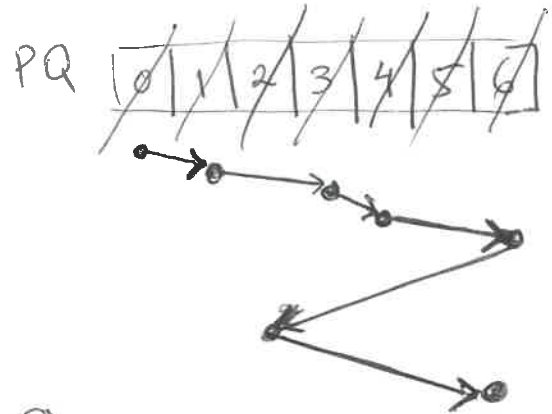
$\boxed{RT = O(n \cdot 2^n)}$ → total RT

- priority queue PQ
- insert() and remove() operations take $O(\lg n)$ for an n-element priority queue

Branch-and-bound with BestFS

Knapsack problem

maxprofit = ~~0~~
~~40~~ 1
~~70~~ 3
90 6



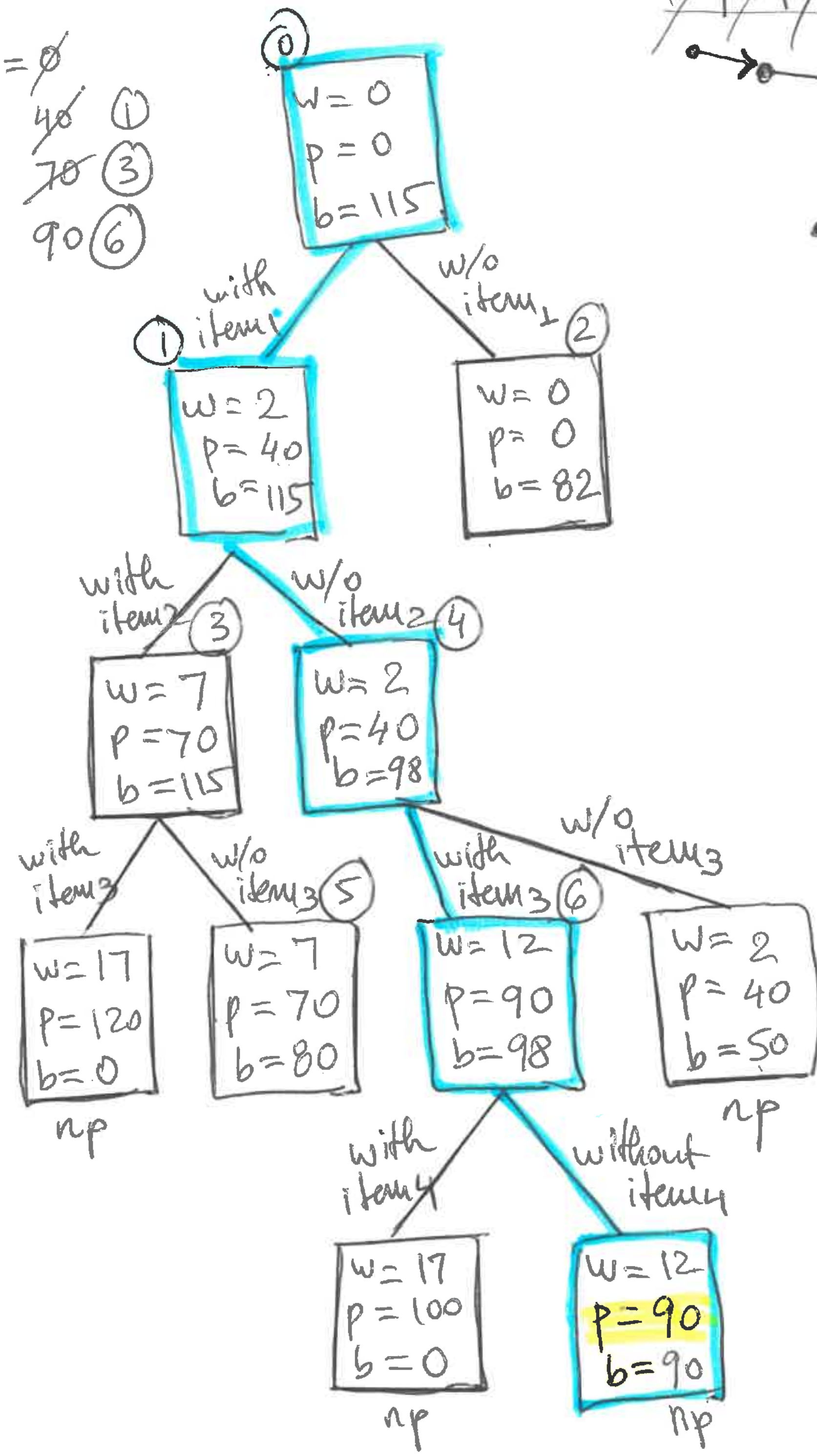
level 0 → 2^0 nodes

level 1 → $\leq 2^1$ nodes

level 2 → $\leq 2^2$ nodes

level 3 → $\leq 2^3$ nodes

level 4 → $\leq 2^4$ nodes



solution = ($item_1$, $item_3$) with weight = 12
profit = 90