Sort by Service time

$O(n \log n)$

زمانبندی سست

Min (Service + wait)

$P_1 : 5$

$P_2 : 4$

$P_3 : 10$

1 , 2 , 3

$5 + 5 + 4 + 5 + 4 + 10 = 33$

2 , 1 , 3

$4 + 4 + 5 + 4 + 5 + 10$

$= 32$

3 , 2 , 1

$10 + 10 + 4 + 10 + 4 + 5 = 43$

زمانبندی با صرفه‌گیری بازدهی و در نظر گرفتن مهلت

| | بازدهی | مهلت |
|---|---|---|
| $P_1$ | 10 | 1 |
| $P_2$ | 8 | 2 |
| $P_3$ | 9 | 1 |

بازدهی

$P_1$ , $P_2$ : 18 ✓

$P_3$ , $P_2$ : 17

{1}

{1,2} → [2,1]  (ثبت)

{2,1,3} → [2,1]

{2,1,4} → [2,1,4]  (ثبت)

{2,1,4,5} ×

6 ×

7 ×

J=[1]

J=K=[2,1]

K=[3,2,1]

K=J=[2,1,4]

K=[5,2,1,4]

K=[5,2,6,4]

| تعلت | ع | تاضا |
|---|---|---|
| 3 | 40 | 1 |
| 1 | 35 | 2 |
| 1 | 30 | 3 |
| 3 | 25 | 4 |
| 1 | 20 | 5 |
| 3 | 15 | 6 |
| 2 | 10 | 7 |

Schedule(n, deadline[ ], profit[ ]) → $O(n^2)$

. Sort by profit → $n\log(n)$

J = [1]

for i = 2 ⟶ n :

$$\sum_{i=2}^{n} (i-1) + i \longrightarrow O(n^2)$$

K = J.append(next(i)) order by deadline

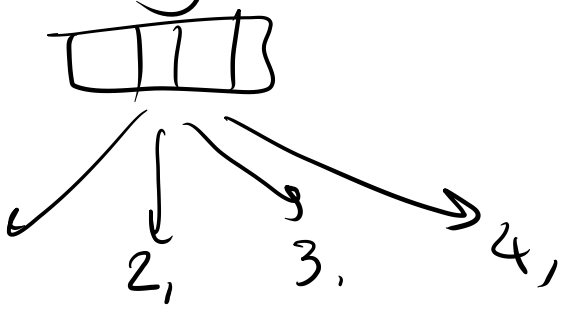
if feasible(K) # میکنن طلب

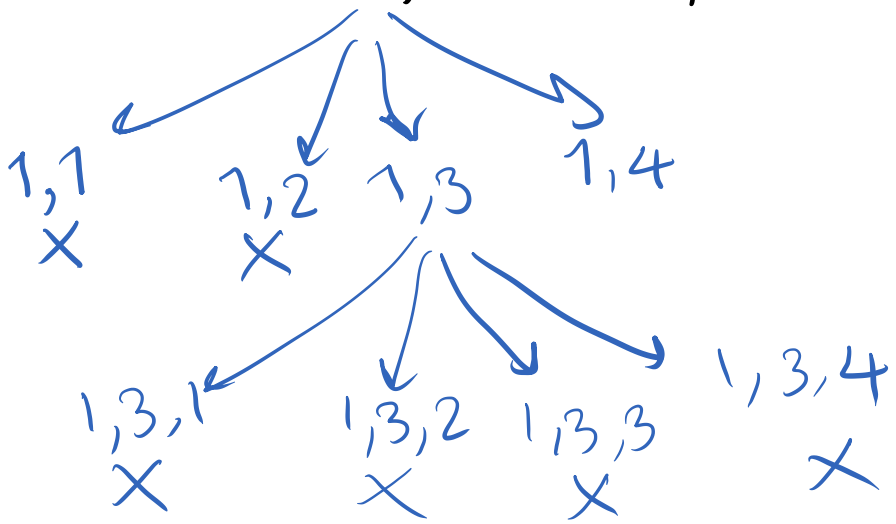
J = K

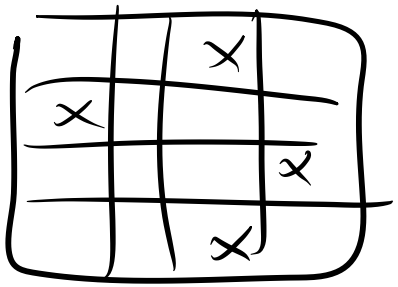
return J

# Backtracking

$n = 4$



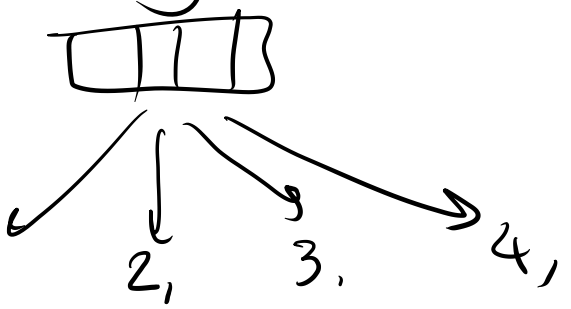



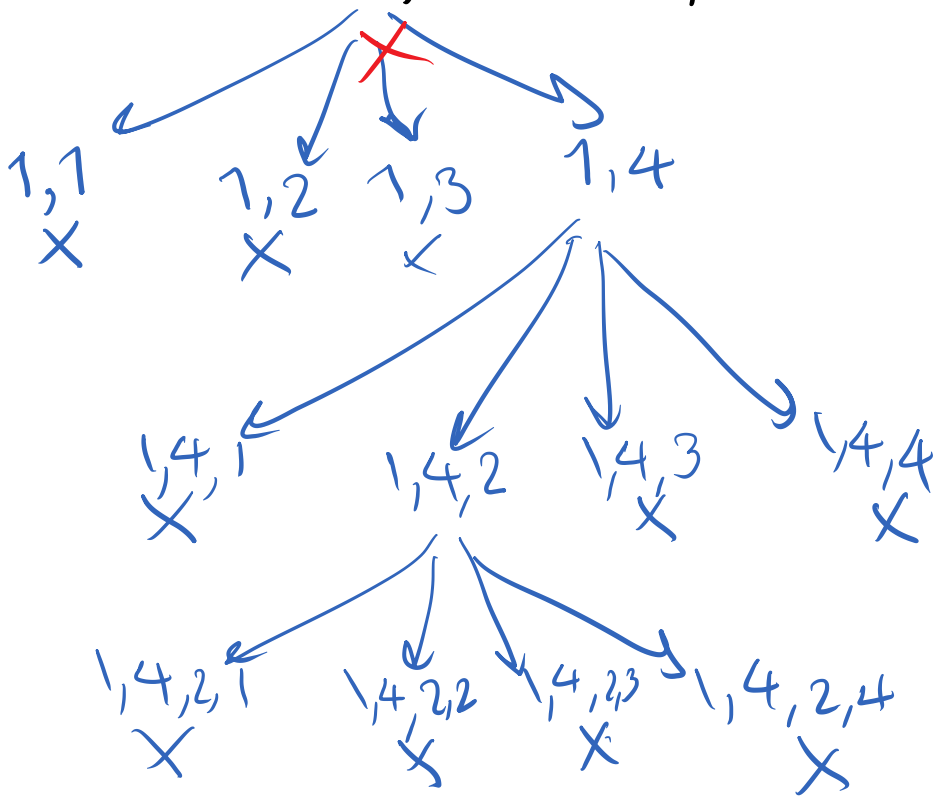



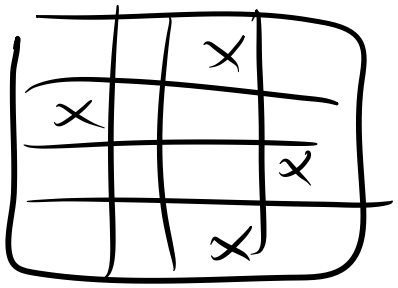
[3, 1, 4, 3]

# Backtracking

$n = 4$



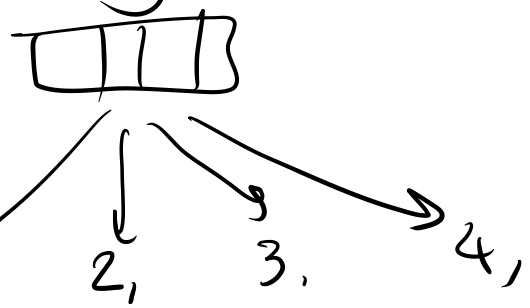



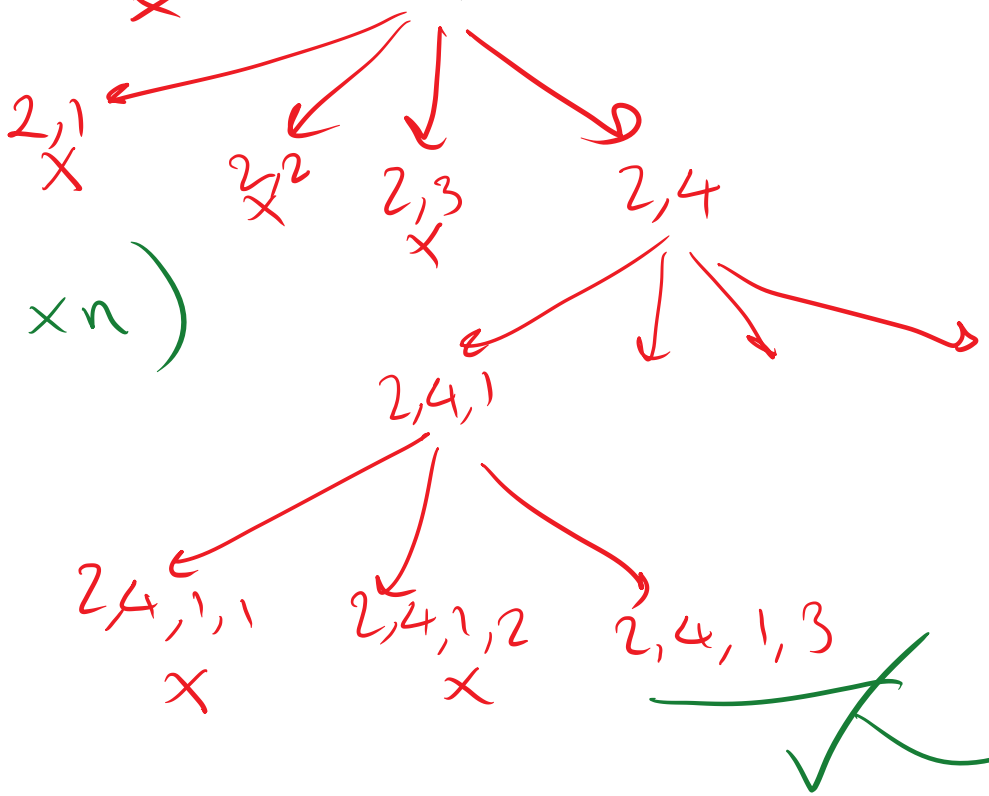



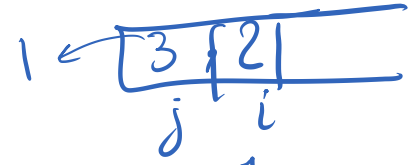
[3, 1, 4, 3]

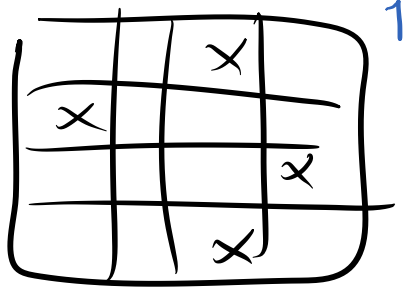
# Backtracking

$n = 4$

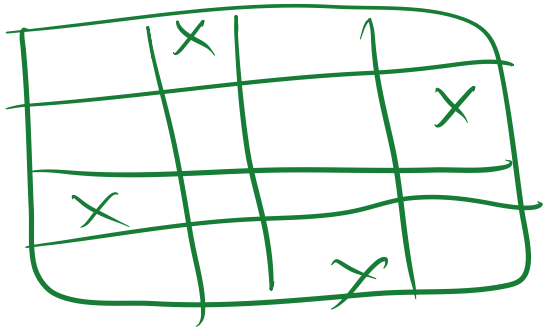





[3, 1, 4, 3]



```
checknode (v):
    if promising(v):
        if is_answer(v)
            return v
        else
            for each child of node v:
                checknode (child)
```

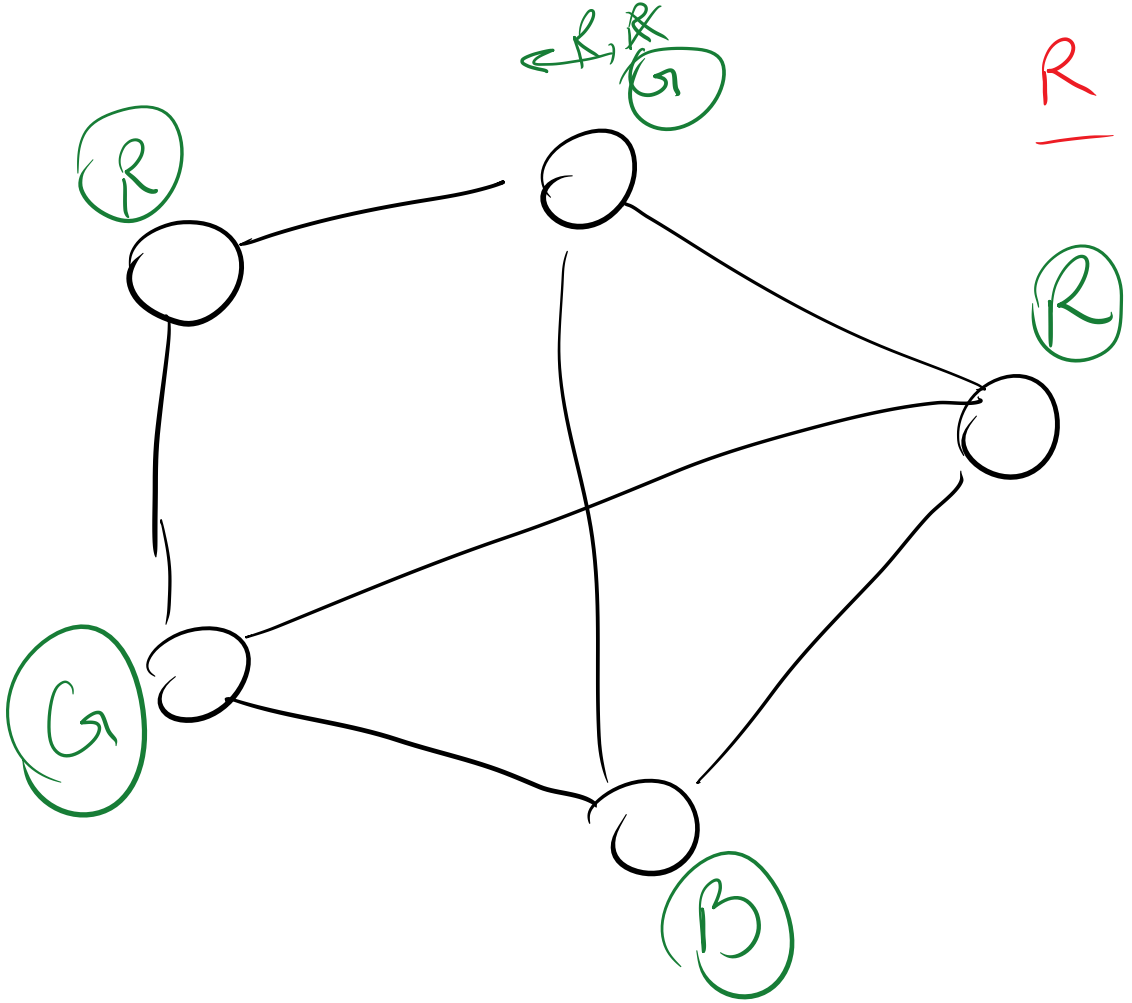
```
N_vazir(cols[i]):
    if promising(cols[i]):
        if i==N:
            return cols

    else
        for j=1→n:
            cols[i+1]=j
            N_vazir(cols[i+1])
```

cols=[ , , , , ]$_N$

```
def promising(cols[i]):
    f = True
    for j = 1 → i-1:
        if cols[j] == cols[i] or
           abs(i-j) == abs(cols[i]-cols[j]):
            return false
    return f
```

| n | Time for first Answer | Time for whole answers |
| --- | --- | --- |
| 4 | — | — |
| ⋮ | ⋮ | ⋮ |
| 20 | — | = |




R G B

رنگ آمیزی گراف:

گراف همبند

با

m رنگ