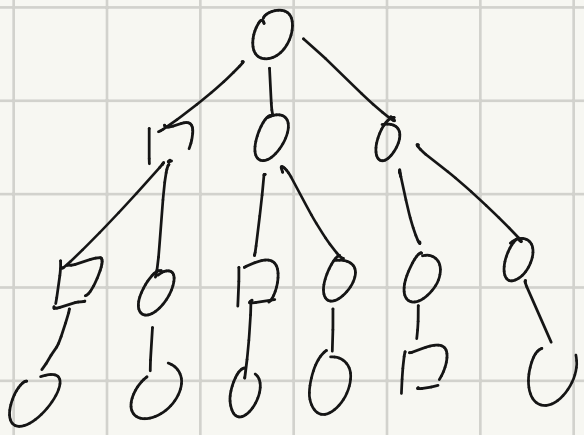
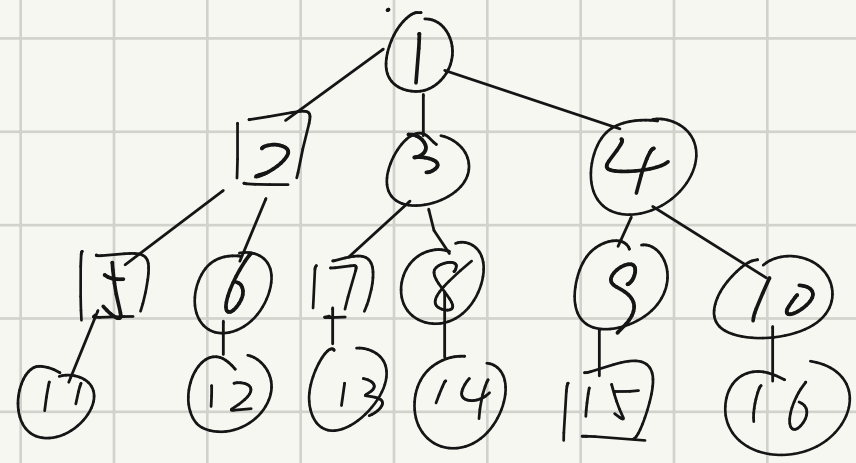
backtracking



O: good

□: bad.



find a good path from root to a leaf.

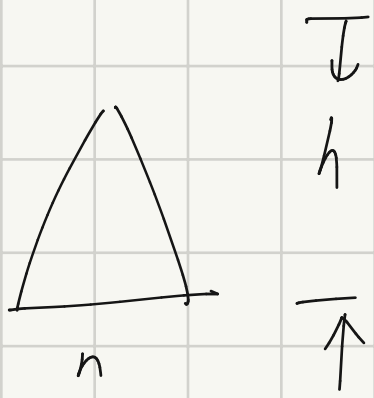
↓

O-O-O-O

DFS or BFS equal in the worst situation

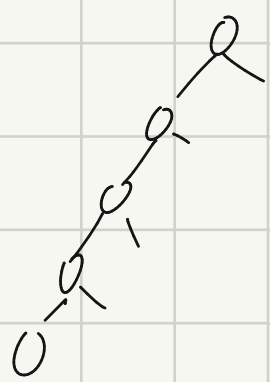
```
dfs (u):        // find a good path from u to a leaf.
1. if u is bad.  => 剪枝.
        return null
2. else     // u is good.
3.      if u is a leaf
4           return u
5       else
6           for each child v of u
7               path = dfs (v)
8               if path ≠ None
9                   return u -> path
```

dfs + pruning = backtracking

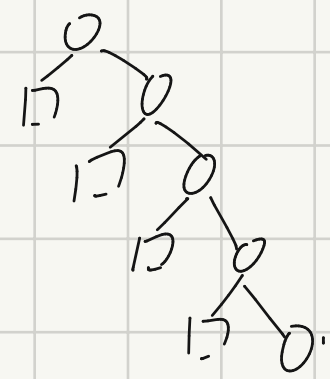


worst case $\Theta(n)$

best case : $O(h)$ (2 situations)



or



fail early

fail prunning

N Queens problem.

Given a $n \times n$ chessboard

find a feasible placement of n queens

no two queens can attack each other

same row

or same col

or digonal

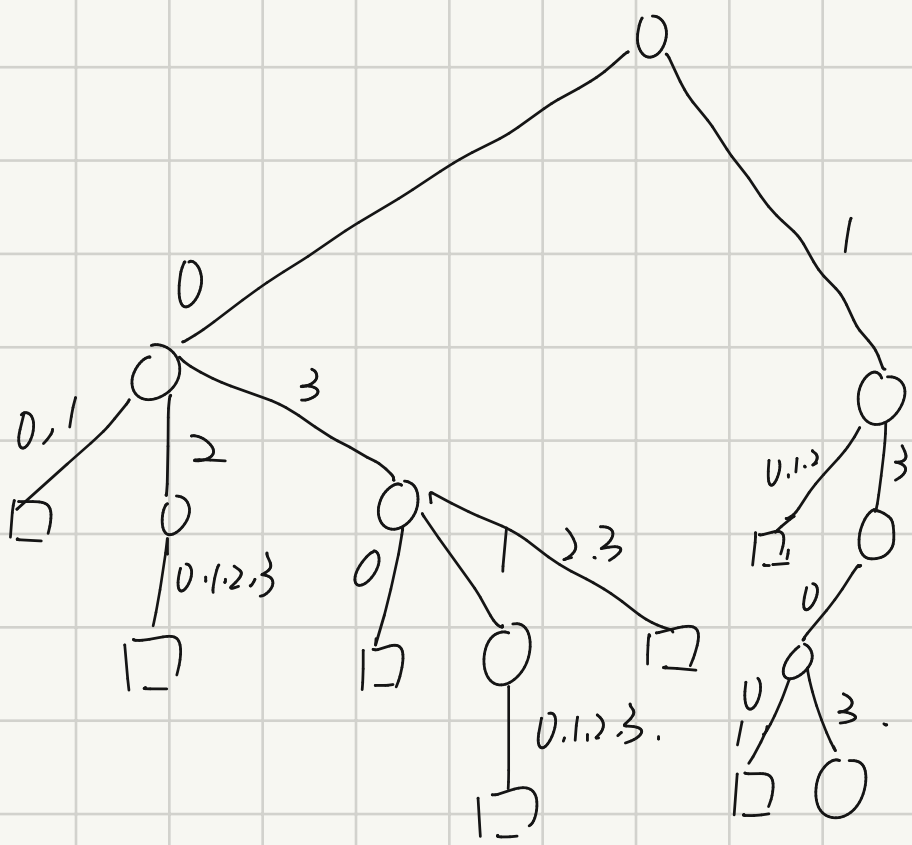
Fact (needn't proved)

1. for $n > 3$, a feasible placement always exist
2. for some special n (prime, $6k+1$)
   efficiently solved
3. for general n.
   NP-hard

bruteforce: $O(n!\,n^2)$

backtrack: $O(n!\ n)$. (worst)

	0	1	2	3
0		Q		
1				Q
2	Q			
3			Q	



Vertice: represent colum.

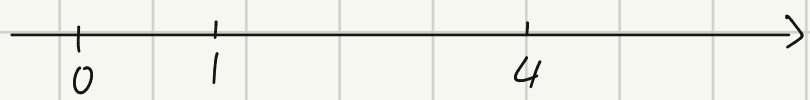
树不用显示地构造，直接搜索即可

```
NQ(n)
    P[i] = -1   for  i=0 to n-1        P reprent Queens' position.
    i=0
    while i≥0  and  i<n
            findgood = false
            for j = P[i]+1  to  n
                if j cannot attack P[0] P[1] . ... P[i-1]
                        P[i] = j
                        findgood = true.
```

```
        if findgood = true
            i = i+1                dfs
        else // findgood = false
            i = i-1                backtracking
    if i == n :
        P is a feasible placement
    if i == -1 :
        No feasible placement
```

Turnpike problem

eg:



$A = \{0, 1, 4\}$ $\qquad D(A) = \{1, 3, 4\}$

$A = \{0, 1, 2\}$ $\qquad D(A) = \{1, 1, 2\}$

multiset, allow duplication.

$A = \{0, 0, 2\}$ $\qquad D(A) = \{0, 2, 2\}$

$$|D(A)| = \frac{|A|^2 - |A|}{2}$$

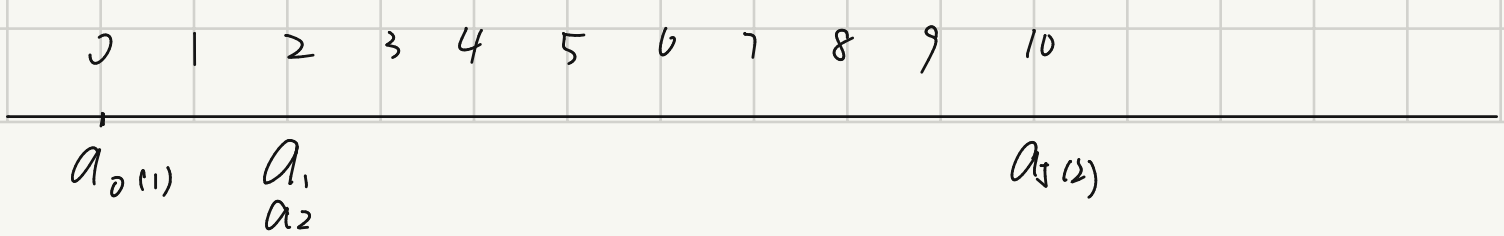
Given D, find $A = \{a_0 \le a_1 \le \cdots \le a_{n-1}\}$, s.t. $D(A) = D$

multiset $\qquad$ assume $a_0 = 0$

$D = \{1, 2, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 8, 8, 10\}$ $\qquad |D| = 15$

$\Rightarrow A = \{a_0 \le a_1 \le \cdots \le a_5\}$

$a_0 \to 0$



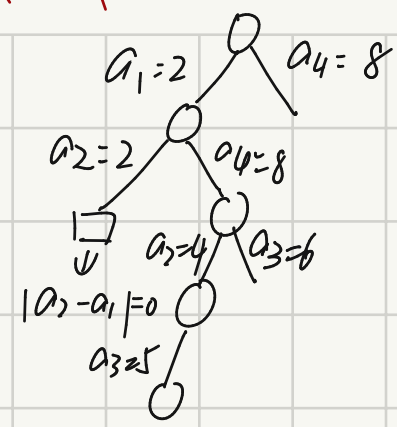
$D = \{1, \cancel{2}, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, \cancel{8}, \cancel{8}, \cancel{10}\}$

8 : $a_5 - a_1$ or $a_4 - a_0$

8 : $a_5 - a_2$ or $a_4 - a_0$

6 : $a_5 - a_2$ or $a_3 - a_0$

5 : $a_5 - a_3$ or $a_3 - a_0$



0	1	2	3	4	5	6	7	8	9	10
$a_{0(1)}$		a_1		a_2	a_3			a_4		$a_{5(2)}$

$D = \{1, \cancel{2}, \cancel{2}, \cancel{2}, \cancel{3}, \cancel{3}, \cancel{4}, \cancel{4}, \cancel{5}, \cancel{5}, \cancel{6}, \cancel{6}, \cancel{8}, \cancel{8}, \cancel{10}\}$

$\therefore A = \{0, 2, 4, 5, 8, 10\}$

1. for every d remaining in D.
   at least one of its endpoints is not determined
2. for maximum d remaining in D.
   at least one of its endpoints is a_0 or a_{n-1}

Input : a multiset D.
Output : a multiset A s.t. $D(A) = D$.

1. $A = \{0, \max(D)\}$
2. $D = D - \{\max(D)\}$
3. TP (D, A)

TP (D, A)

$O(1)$ 1. if D is empty

```
                2.      return true
maxtime         3.  d = max(D)
2X              4.  for a* = d or max(A) - d.
O(n)            5.      Δ = { dist(a*, a) : a ∈ A }
n.findKey       6.      if Δ ⊆ D.
n.deltime       7.          D := D - Δ
O(1)            8.          A := A ∪ {a*}
O(1)            9.          if TP(D, A)
                10.             return true
n.insertion     11          else
time            12              D := D ∪ Δ
O(1)            13.             A := A - {a*}
                14  return false
```

∴ $O(n)$ + max time + n findkey + n del + n ins

AVL Tree

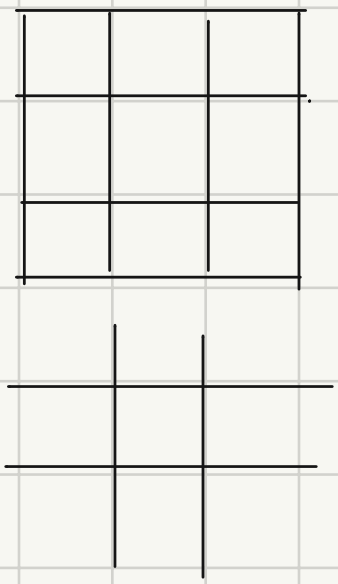
time per node $O(n \log n)$

worse case : $O(2^n)$ nodes (rare)

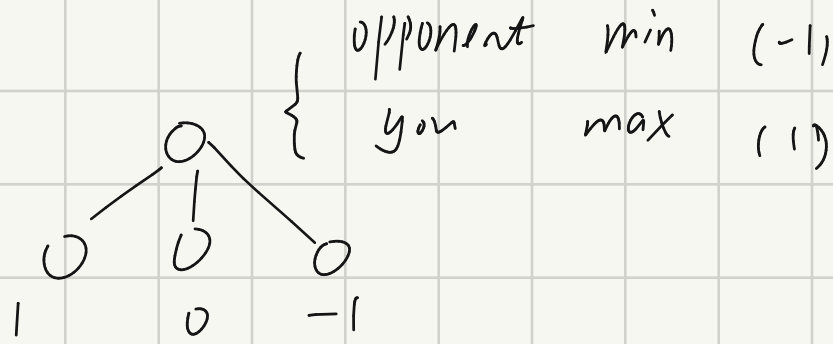
best case : $O(n)$ nodes (most instances)

Game

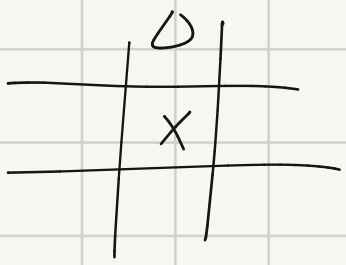
Tic-tac-toe



1 0 -1



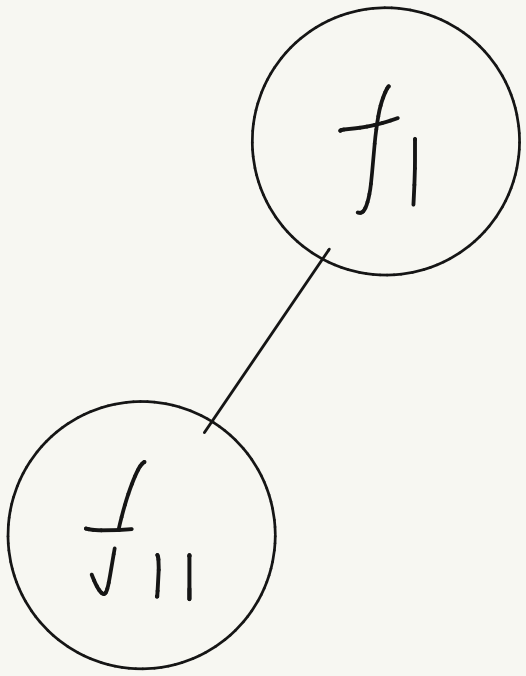
$$f(p) = \underbrace{W_{you}}_{\#\ \text{potential wins}} - W_{oppo}$$



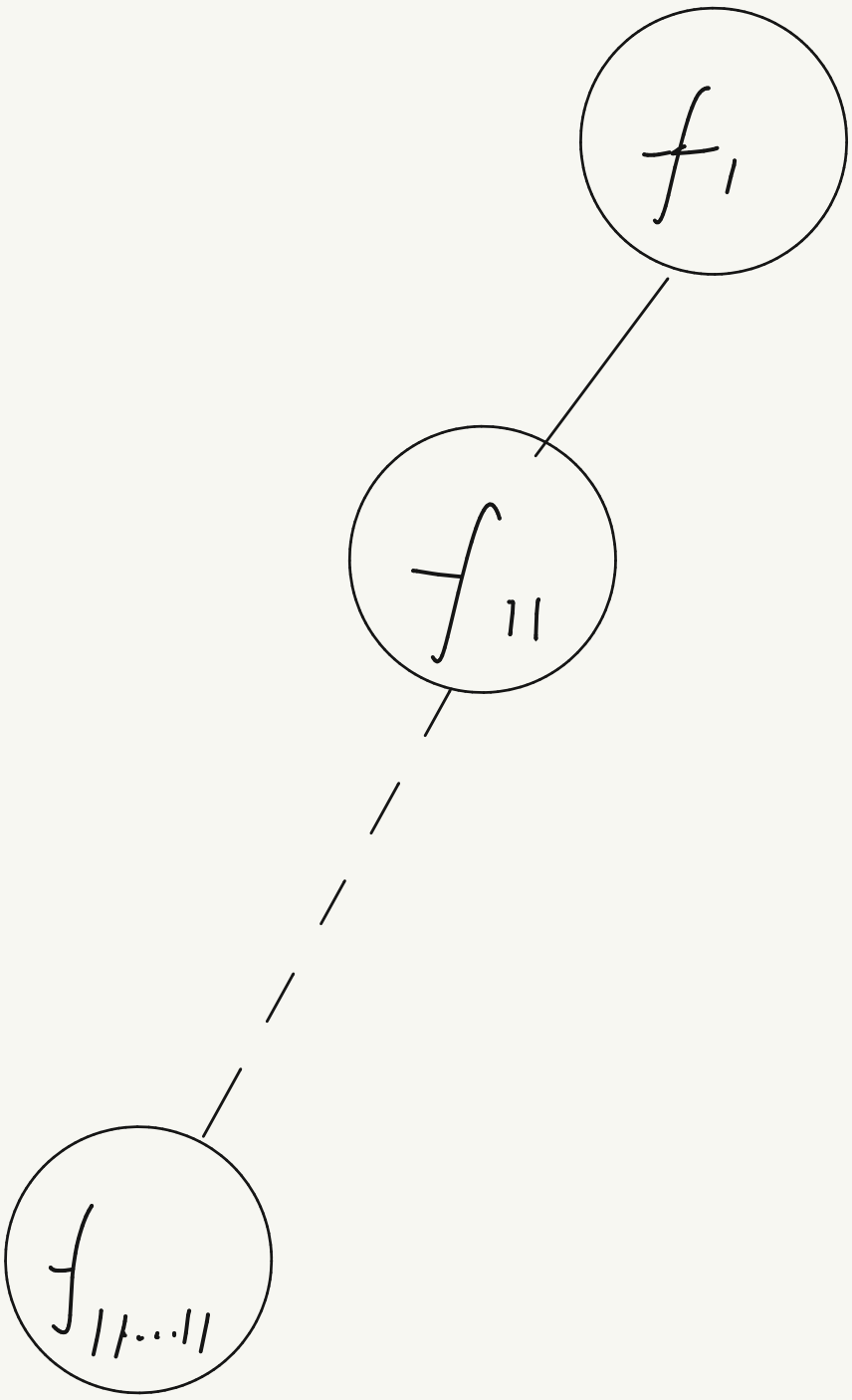
$W_{you} = 6$

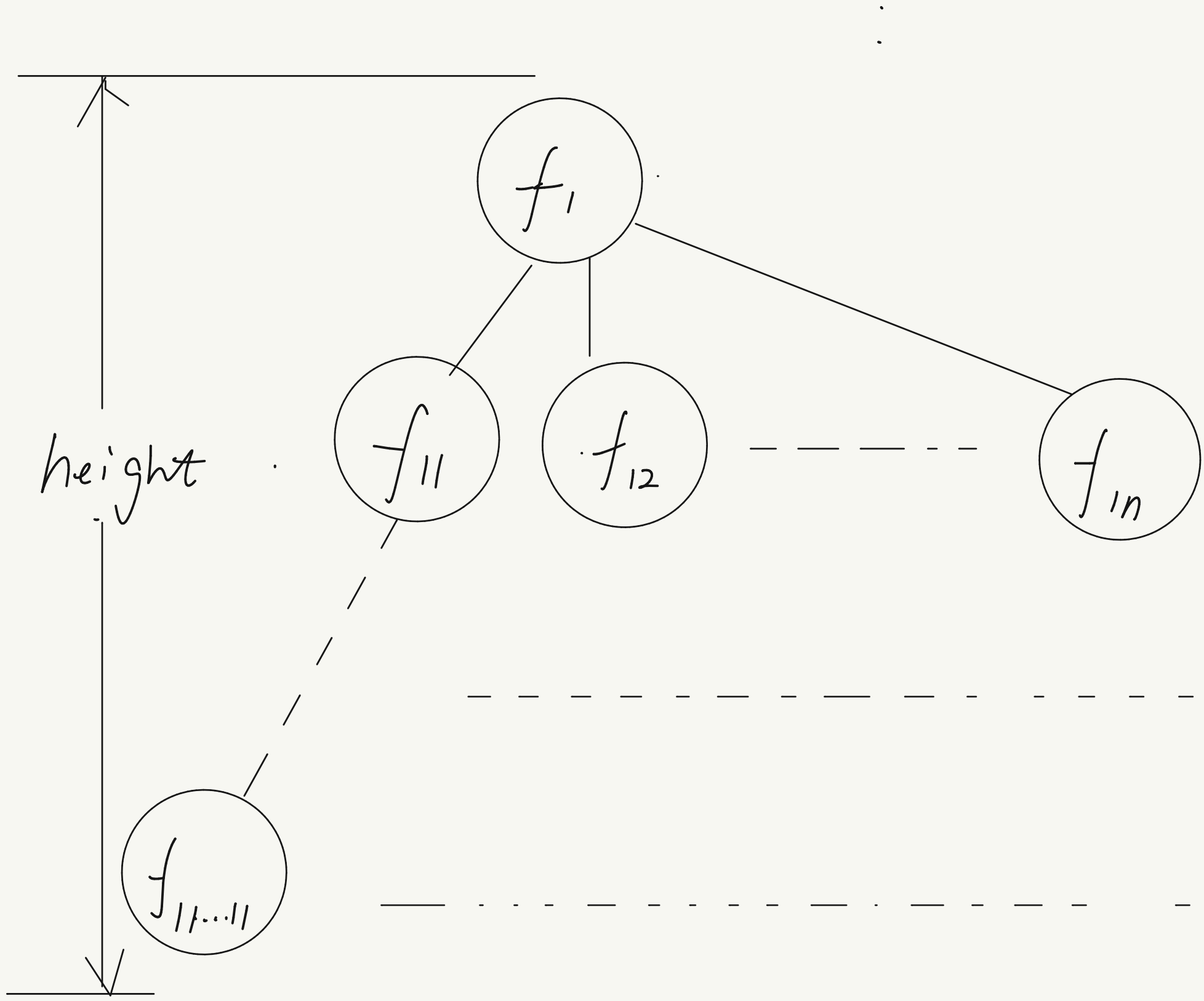
$W_{opponent} = 4$

$f(p) = 2$


f_1
f_{11}


f_1


f_1
f_{11}
$f_{11\ldots11}$



max-num = height = max length of all paths