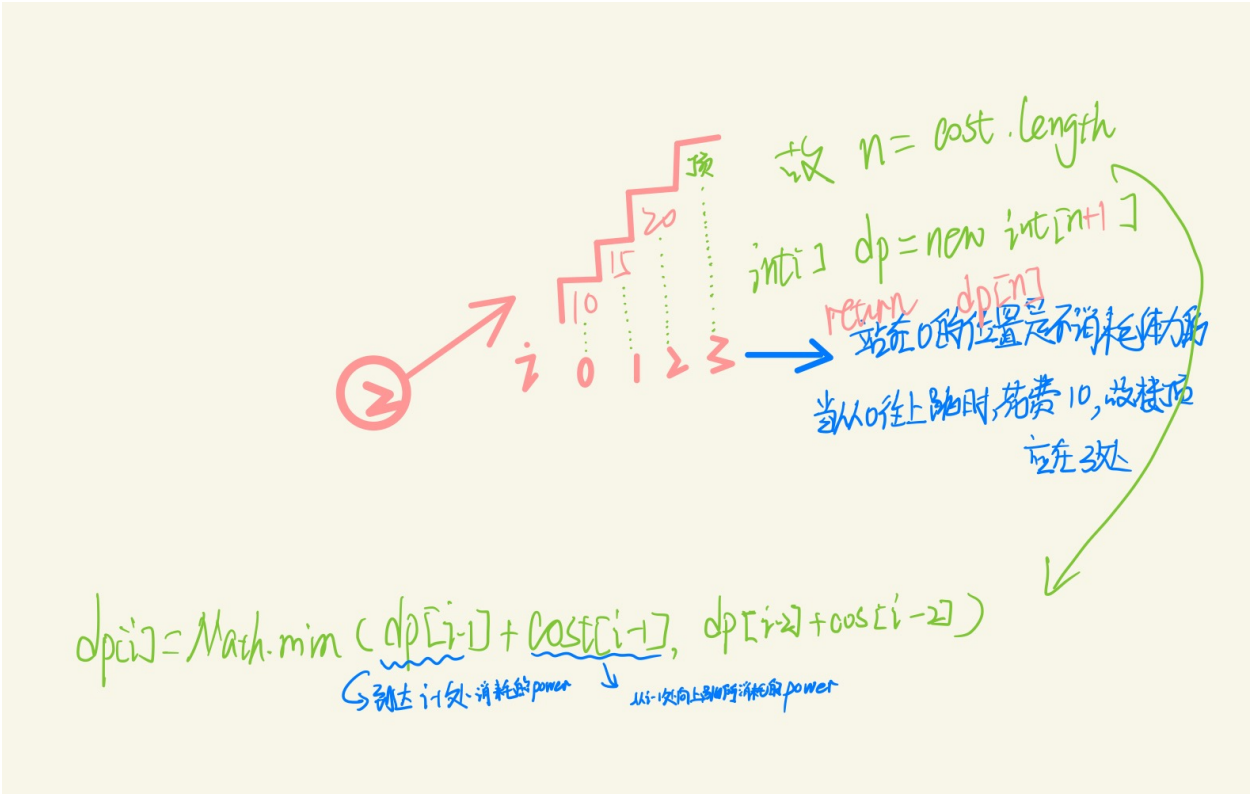
动态规划

使用最小花费爬楼梯

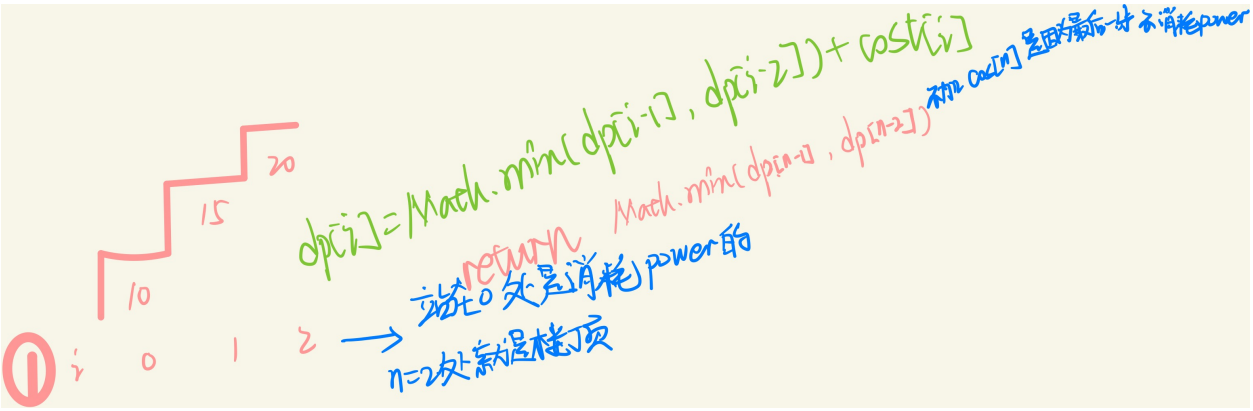
#第一步不消耗体力




```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
    int n=cost.length;
    int[] dp=new int[n+1];//到达第i个台阶需要花费的最少体力为dp[i]
    dp[0]=0;//第0或1步不需要耗费体力（第一步）
    dp[1]=0;
    for(int i=2;i<=n;i++){
    dp[i]=Math.min(dp[i-1]+cost[i-1],dp[i-2]+cost[i-2]);
    }
    //最后一步需要耗费体力
    return dp[n];
```

```
    }
}
```

#第一步消耗体力




```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
    int n=cost.length;
    int[] dp=new int[n];//到达第i个台阶需要花费的最少体力为dp[i]
    dp[0]=cost[0];//第0或1步需要耗费体力
    dp[1]=cost[1];
    for(int i=2;i<n;i++){
    dp[i]=Math.min(dp[i-1],dp[i-2])+cost[i];
    }
    //最后一步不需要耗费体力，故不再➕cost[n]
    return Math.min(dp[n-1],dp[n-2]);
    }
}
```