[86 74 60 70 80 90 100 110 30]

a+b = tar

duplicate

tar → 160

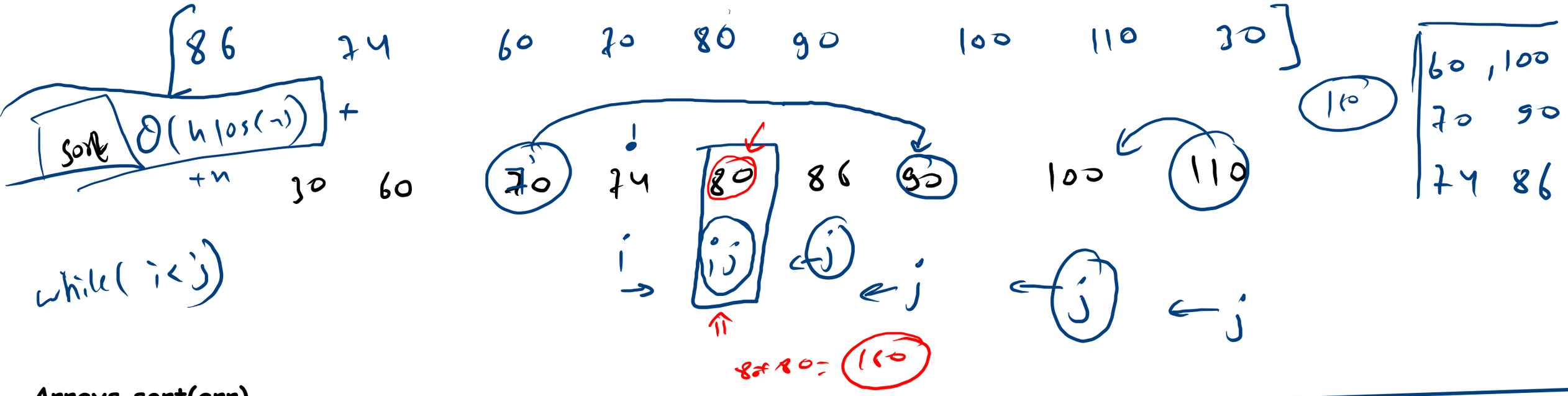
74+86 = 160

60 +100 = 160

70 +90 = 160

h^2

a+b = tar



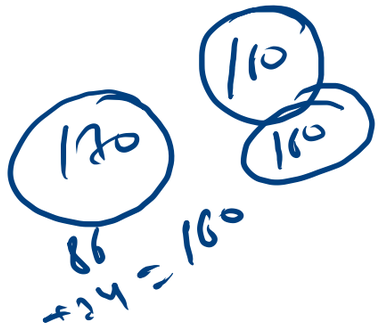
Arrays.sort(arr)

sum = ar[i] + ar[j]

140

170

160



sum < tar

140 < 160

i ++

sum > tar

170 > tar

j --

sum = tar

sum = tar

i ++

j --

i ++ , j --

[10 20 30 40 50]

k

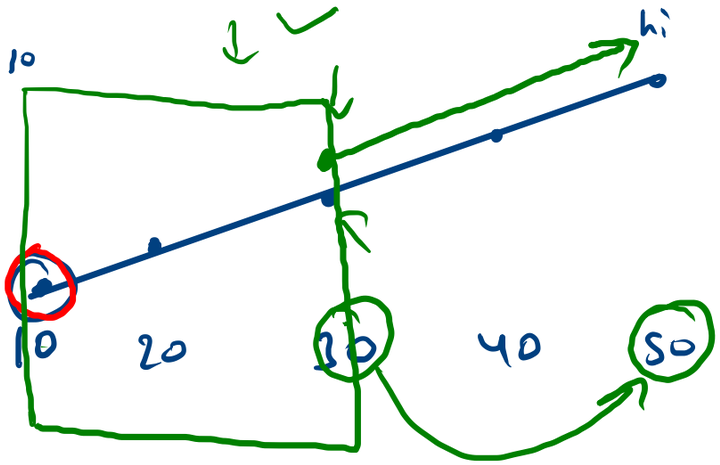
smallest → pivot

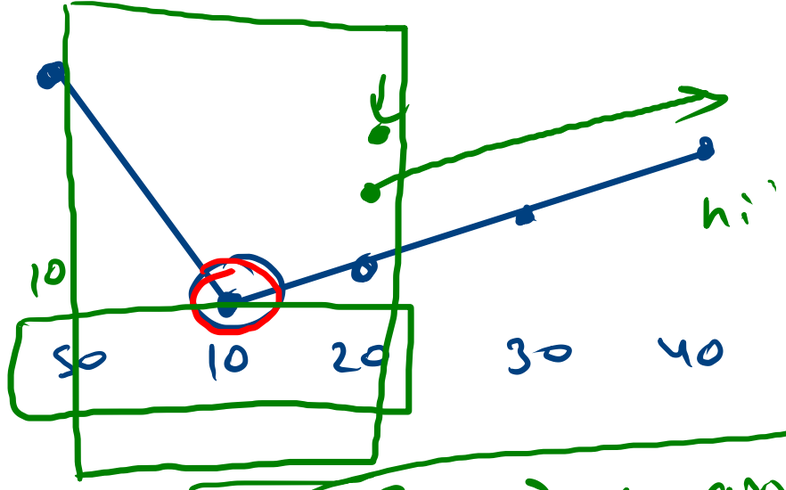
k=0

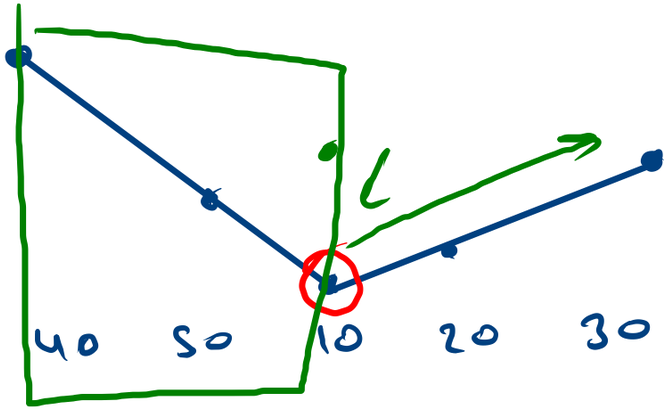
[10 20 30 40 50]

[50 10 20 30 40]

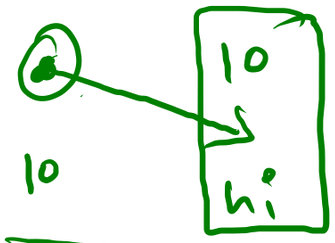
[40 50 10 20 30]





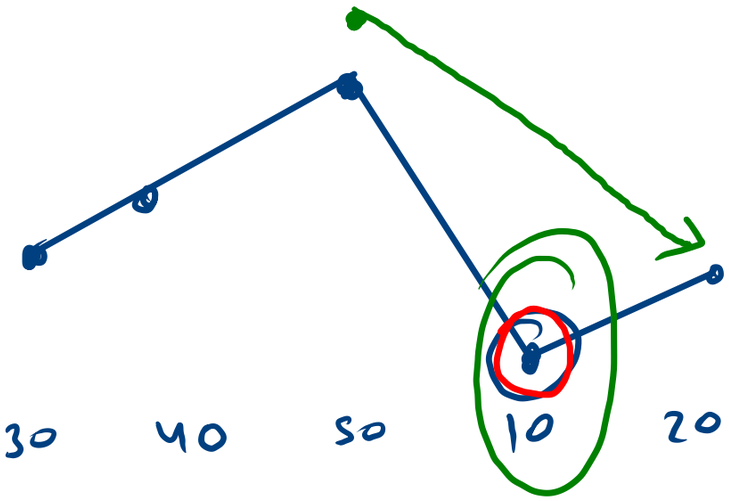


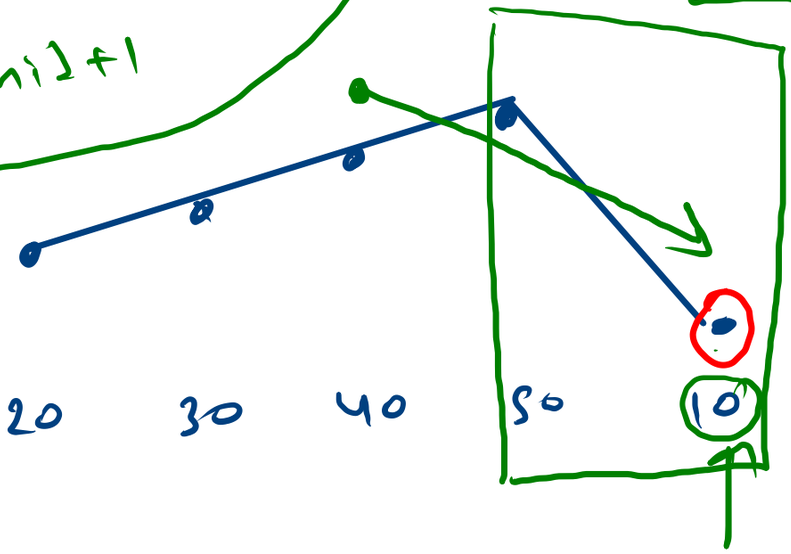

```
arr[mid] < arr[hi]
    hi = mid
else  lo = mid+1
```



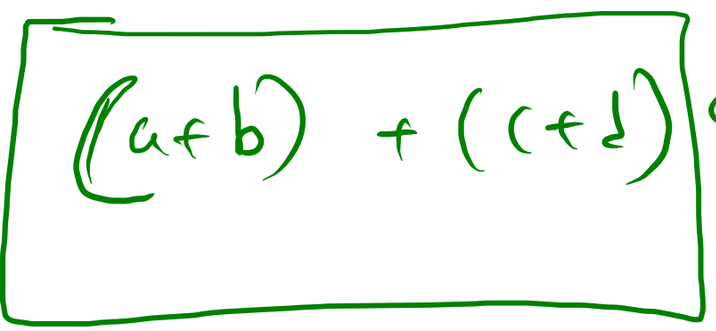
min → O(n)

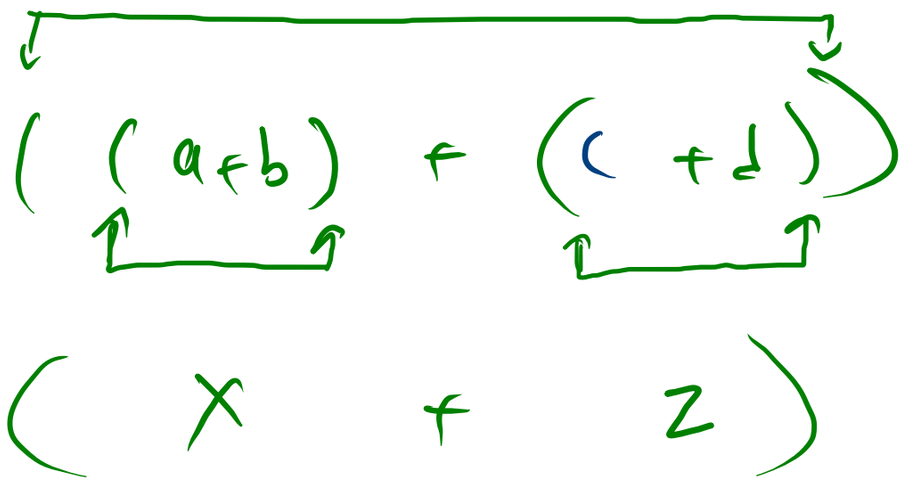
O(log₂(n)) ← BS

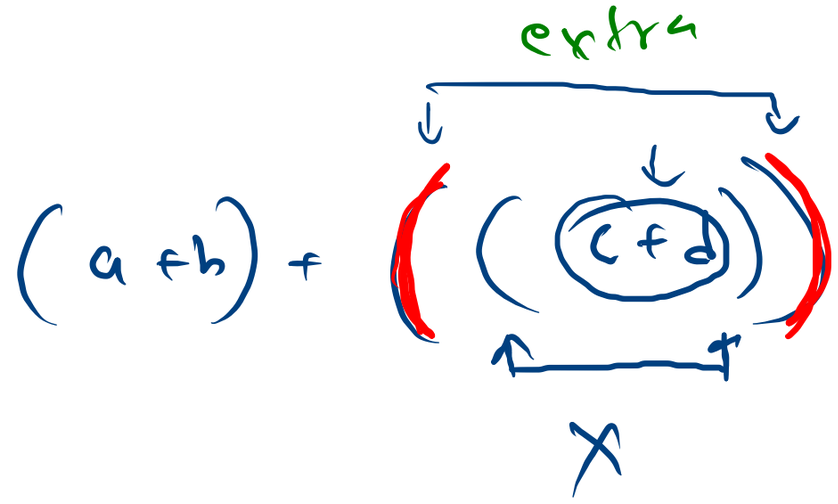


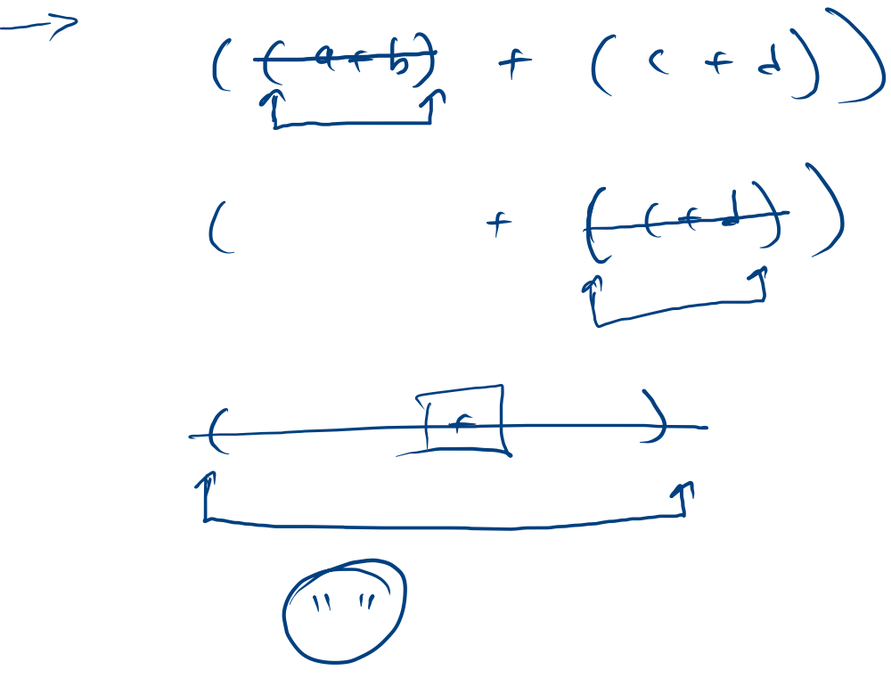


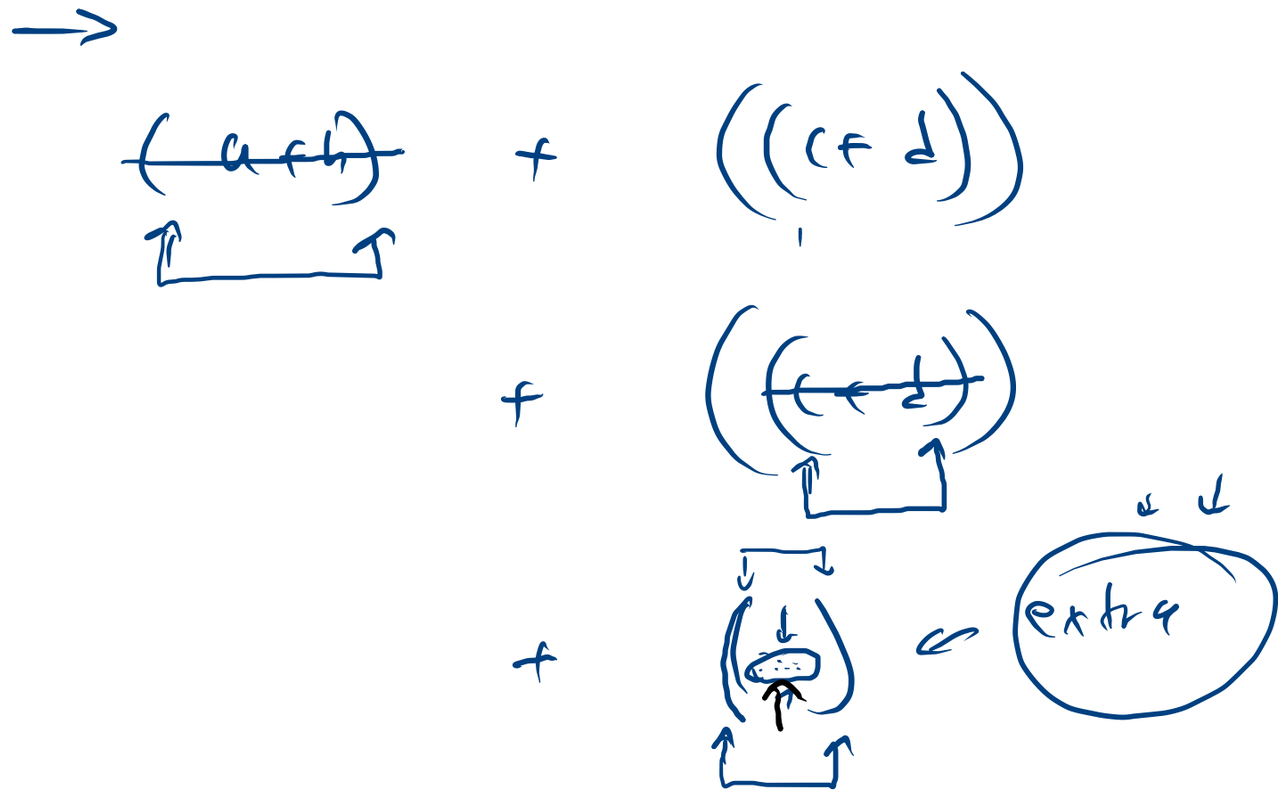
((a + b) + (c + d)) -> false
(a + b) + ((c + d)) -> true

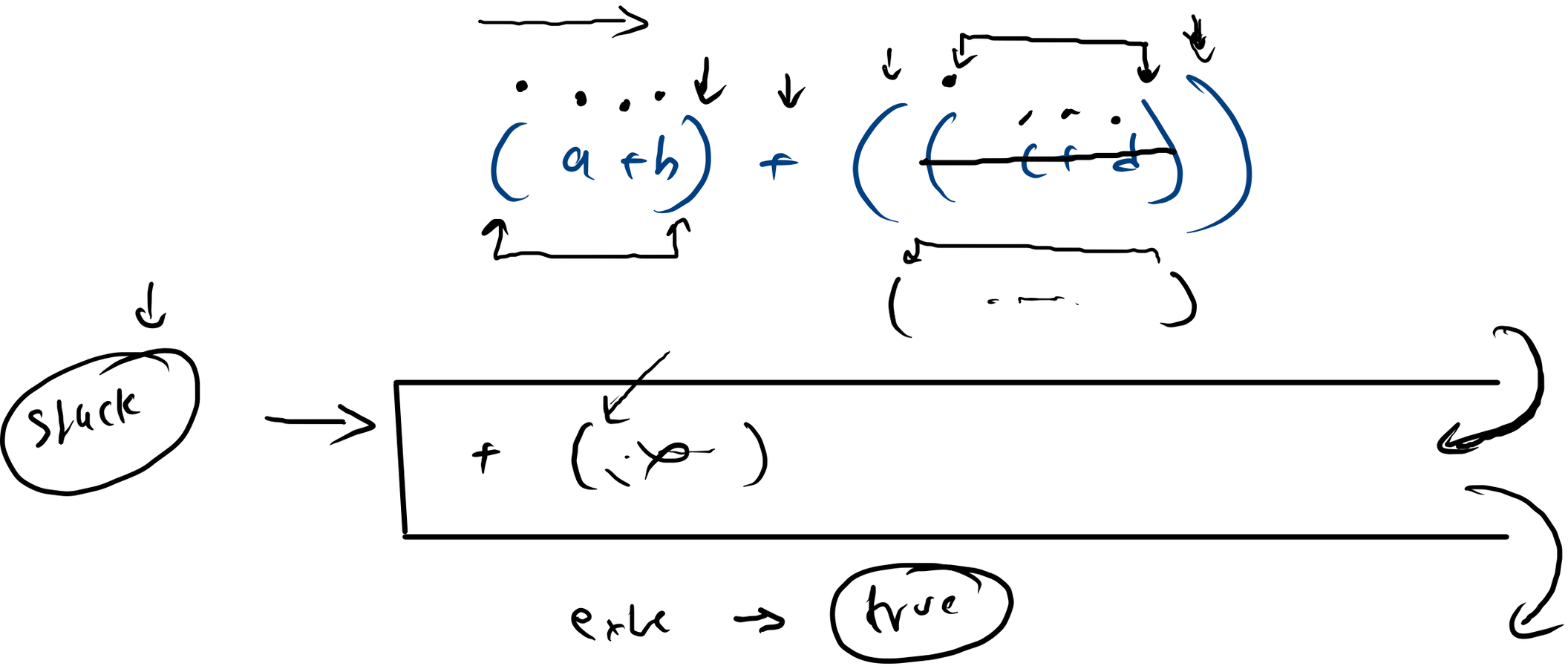






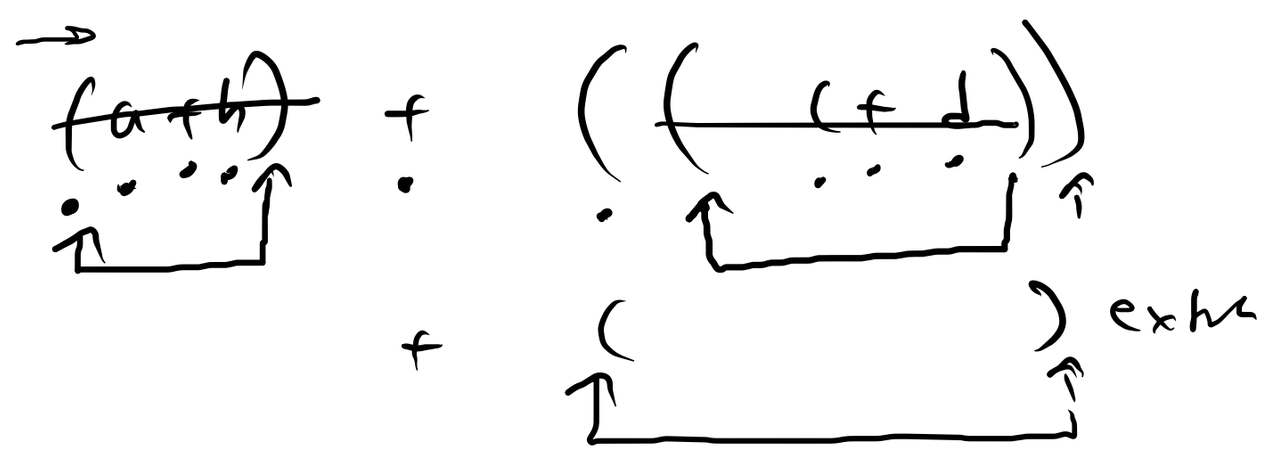


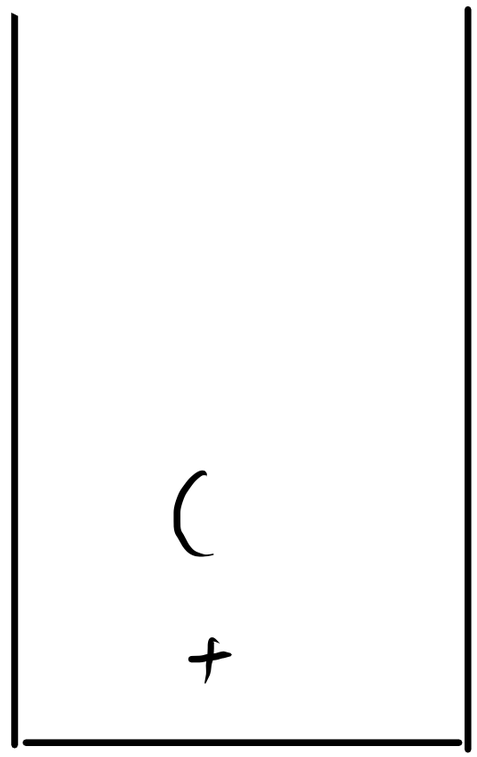

extra


(a + b) + ((c + d))
( )
stack
+ ( )
exit
true

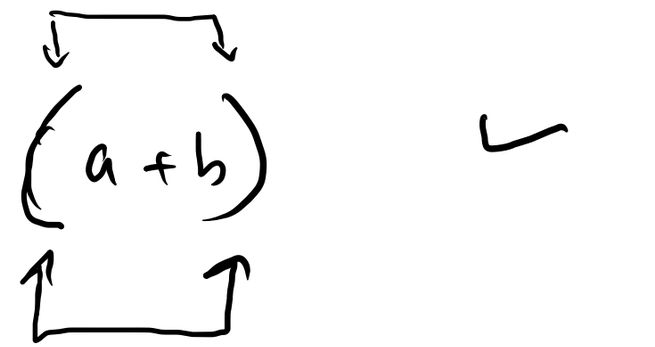
```java
for(int i=0;i<s.length();i++){
    char ch = s.charAt(i);

    if(ch != ')'){
        st.push(ch);
    }else{
        if(st.peek() == '('){
            System.out.println('true');
            return;
        }else{
            while(st.peek() != '('){
                st.pop();
            }
            st.pop();
        }
    }
}
System.out.println('false');
```



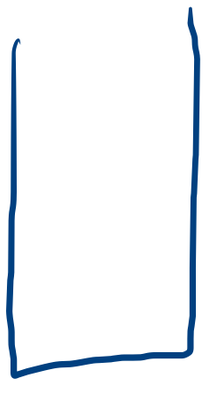


[(a + b) + {(c + d) * (e / f)}] -> true
[(a + b) + {(c + d) * (e / f)]} -> false
[(a + b) + {(c + d) * (e / f)} -> false
([(a + b) + {(c + d) * (e / f)}] -> false



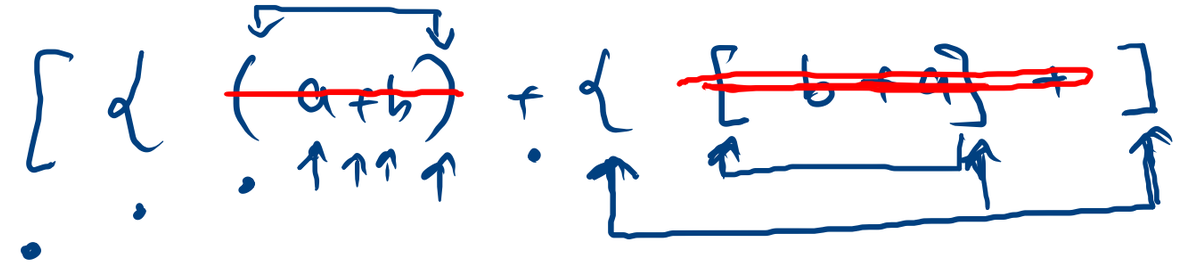
a + b)

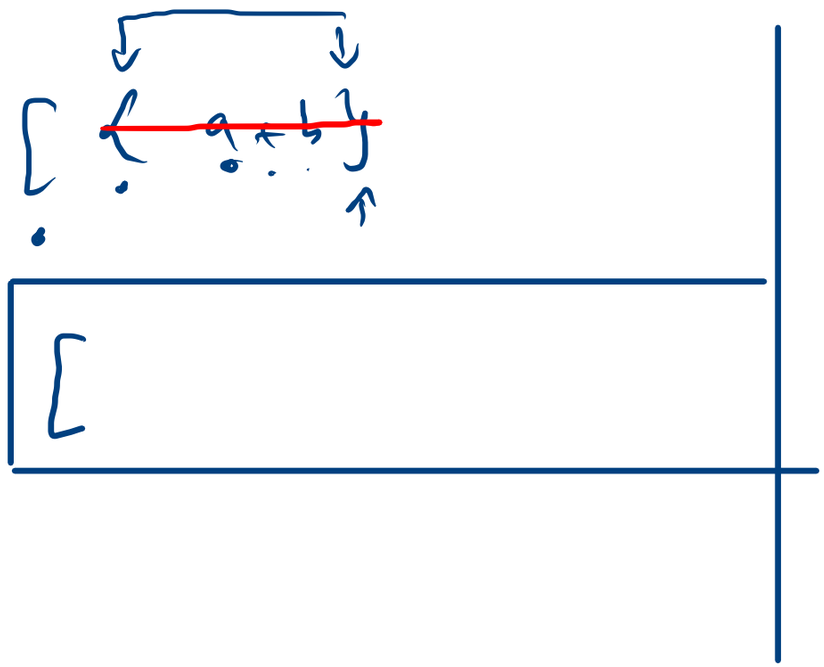
a + b)

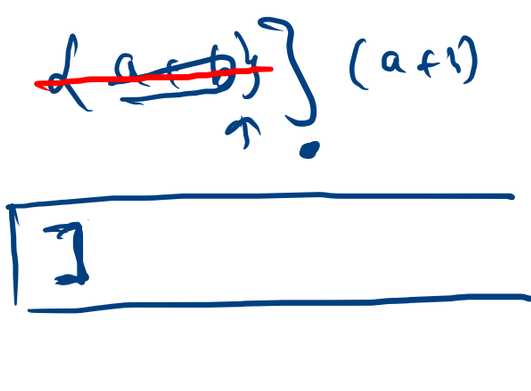


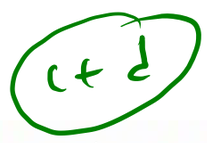
```java
for(int i=0;i<s.length(); i++){
    char ch = s.charAt(i);
    if(ch == '(' || ch == '{' || ch == '['){
        st.push(ch);
    }else if(ch ==')'){
        if(st.size() > 0 && st.peek() == '('){
            st.pop();
        }else{
            ok = false;
            break;
        }
    }else if(ch ==']'){
        if(st.size() > 0 &&st.peek() == '['){
            st.pop();
        }else{
            ok = false;
            break;
        }
    }else if(ch =='}'){
        if(st.size() > 0 &&st.peek() == '{'){
            st.pop();
        }else{
            ok = false;
            break;
        }
    }
}
if(ok && st.size() == 0){
    System.out.println("true");
}else{
    System.out.println("false");
}
```

ok = true







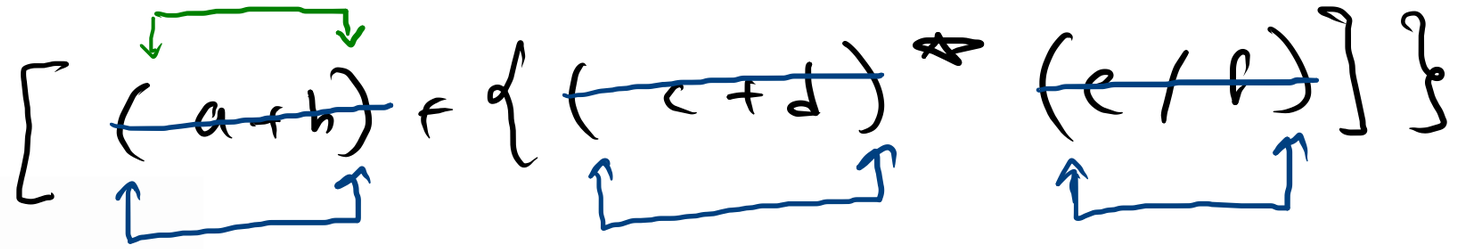


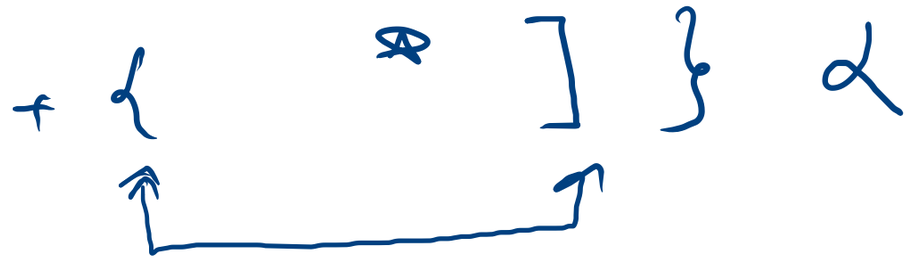
[(a + b) + {(c + d) * (e / f)}] -> true

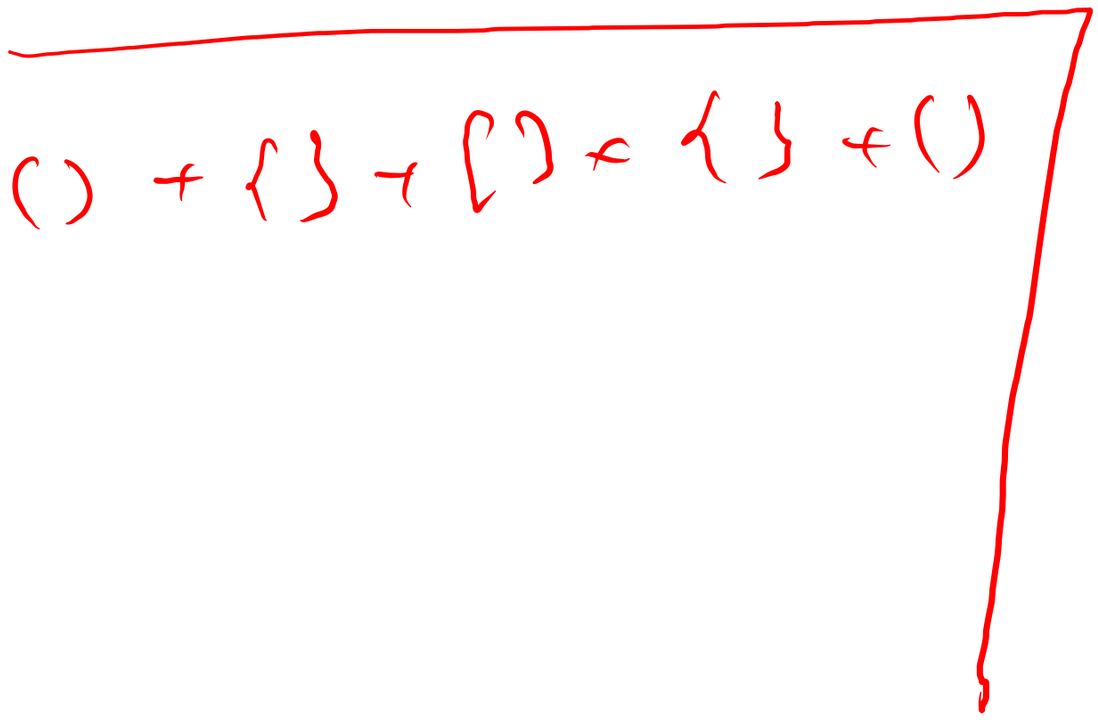
[(a + b) + {(c + d) * (e / f)]} -> false

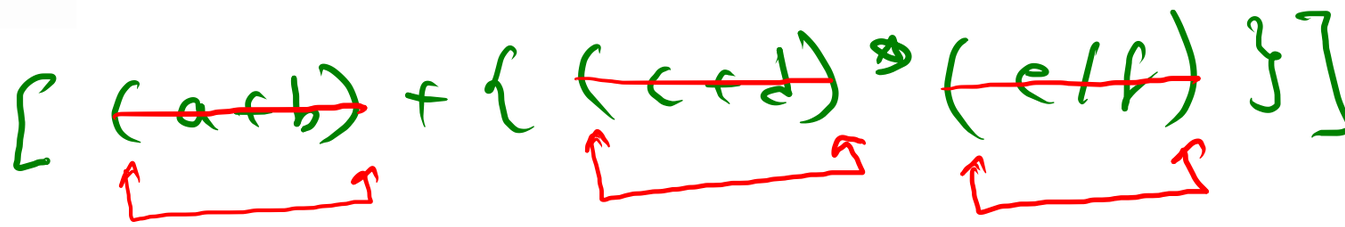
[(a + b) + {(c + d) * (e / f)} -> false

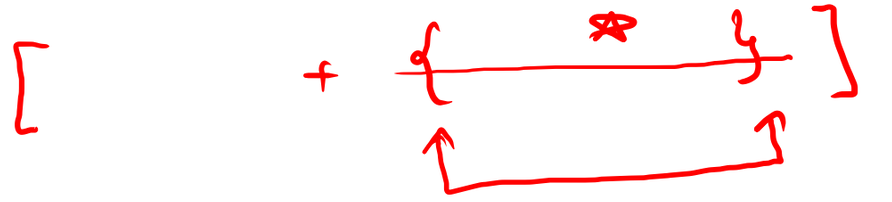
([(a + b) + {(c + d) * (e / f)}] -> false

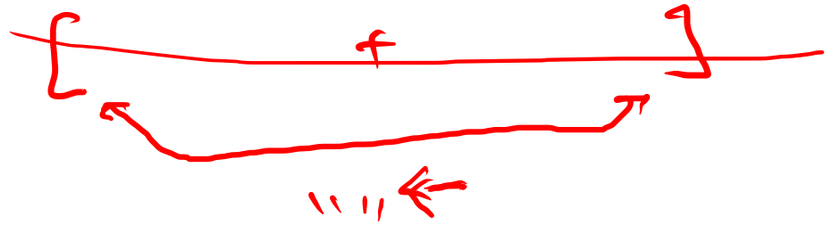










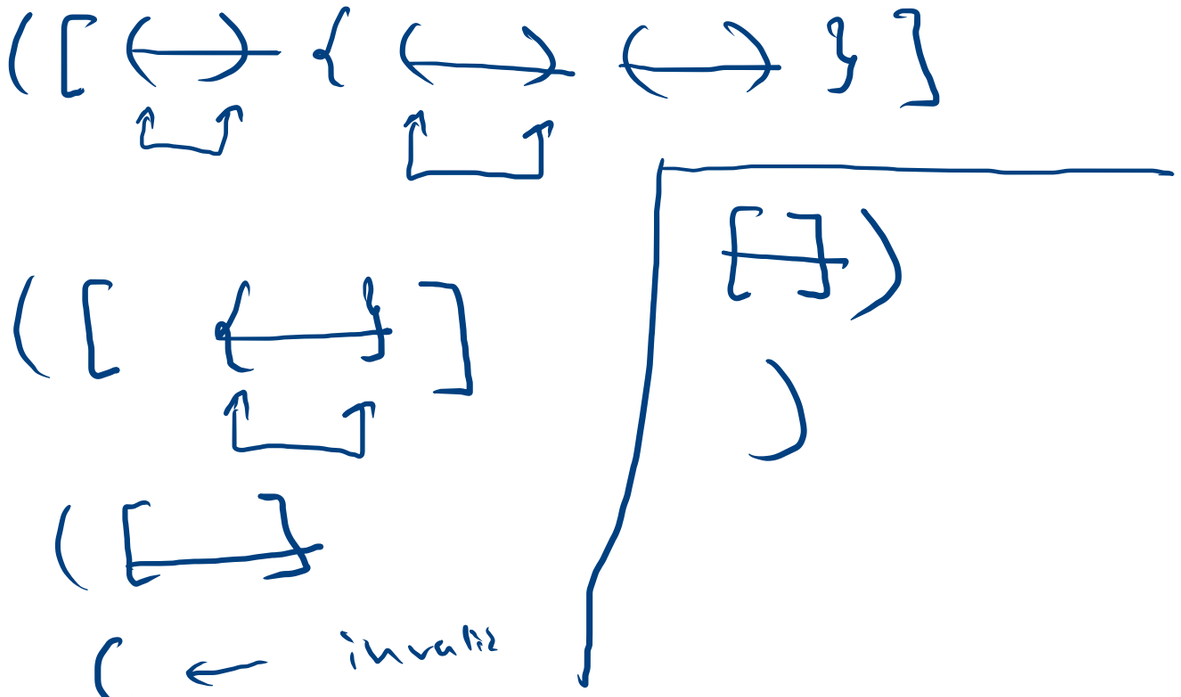


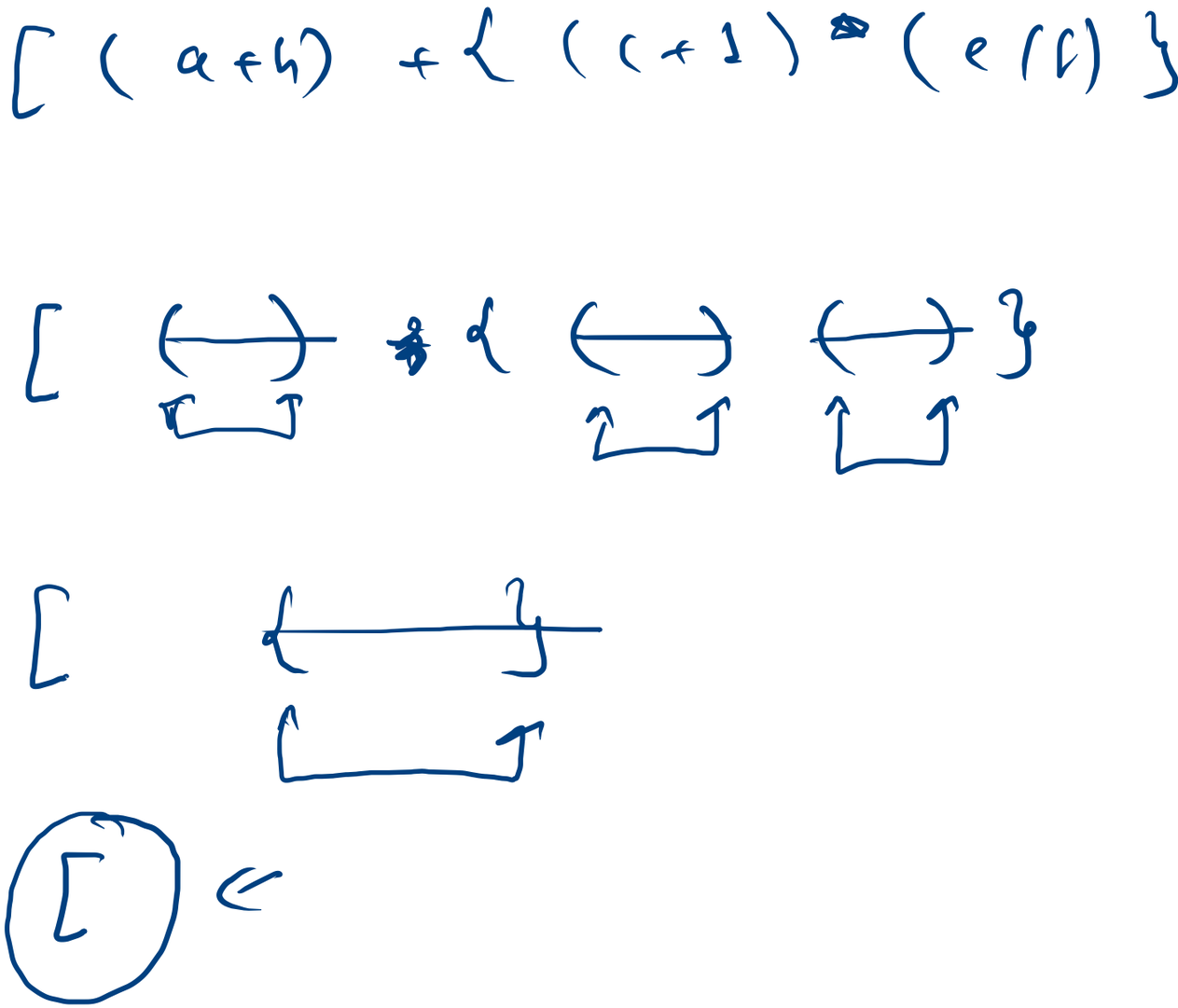
[(a + b) + {(c + d) * (e / f)}] -> true

[(a + b) + {(c + d) * (e / f)]} -> false

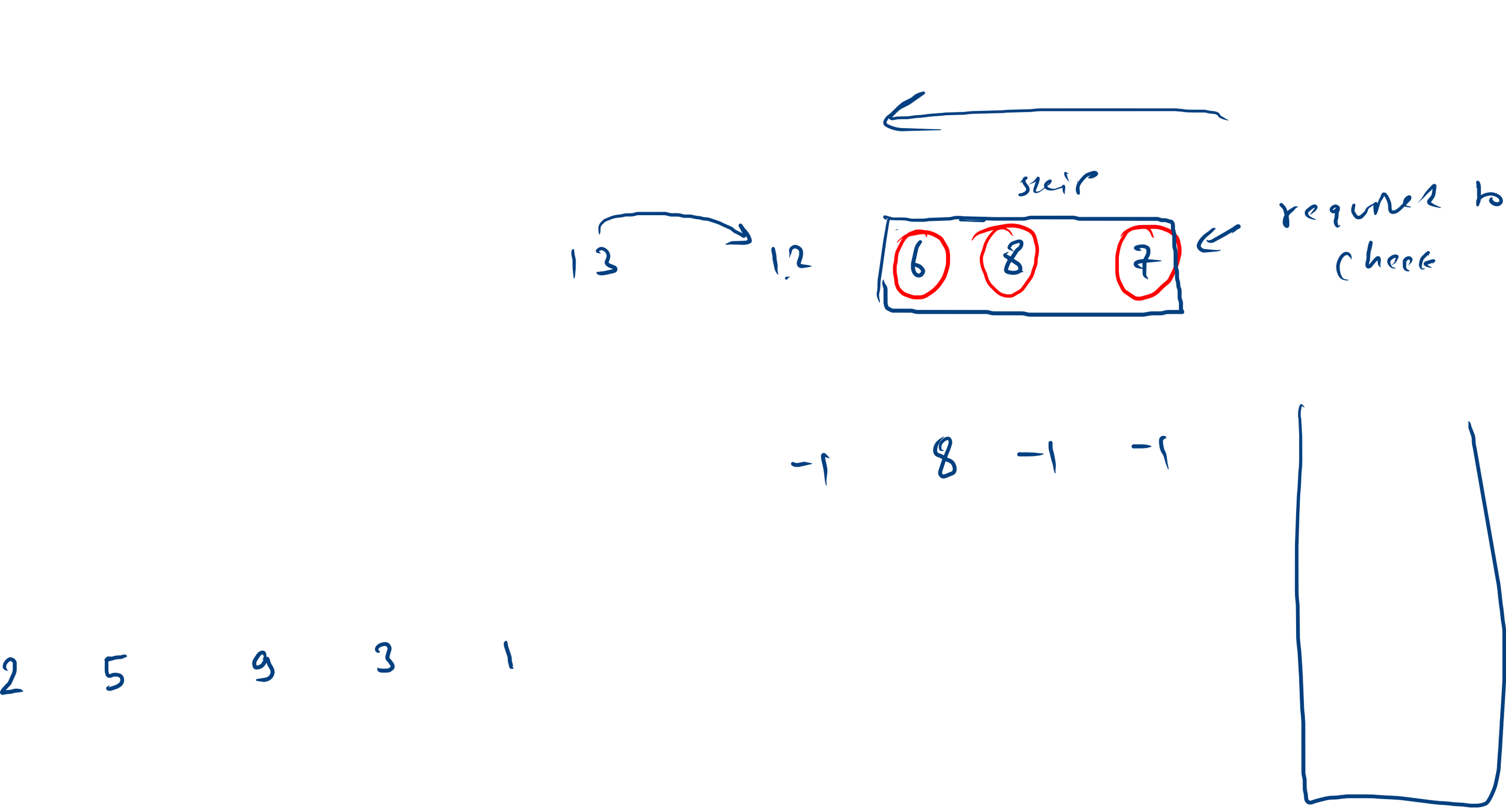
[(a + b) + {(c + d) * (e / f)} -> false •

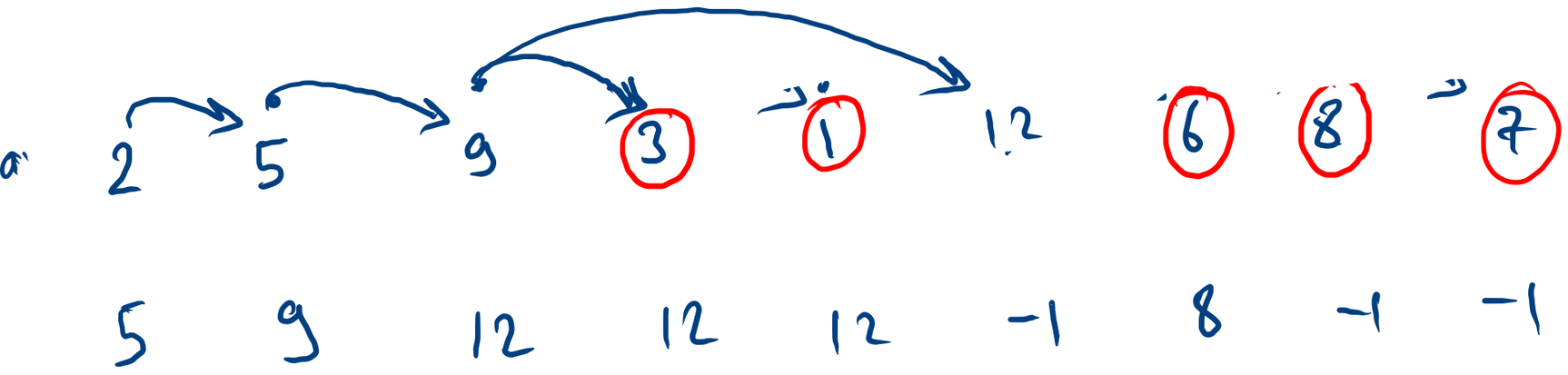
([(a + b) + {(c + d) * (e / f)}] -> false



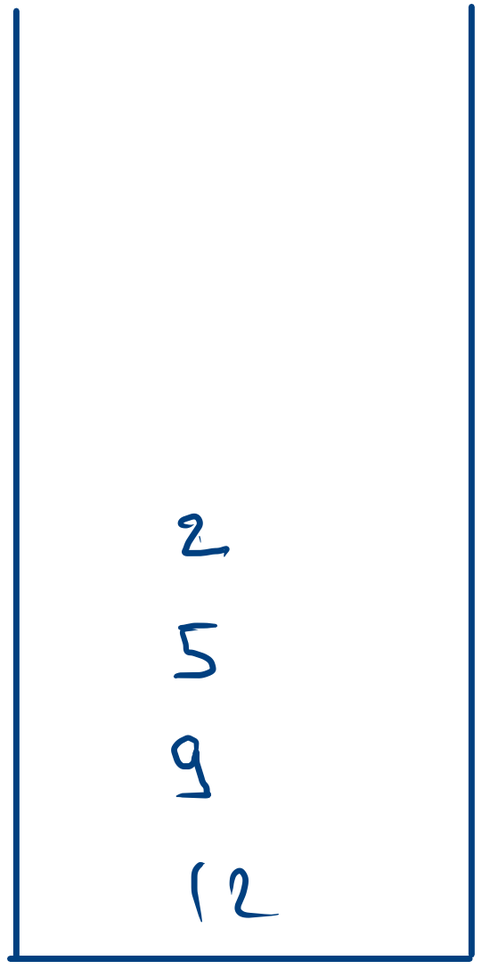


	2	5	9	3	1	12	6	8	7
	•								
nqx	5	9	12	12	12	-1	8	-1	-1


13
12
stair
6
8
7
-1
8
-1
-1
2
5
9
3
1

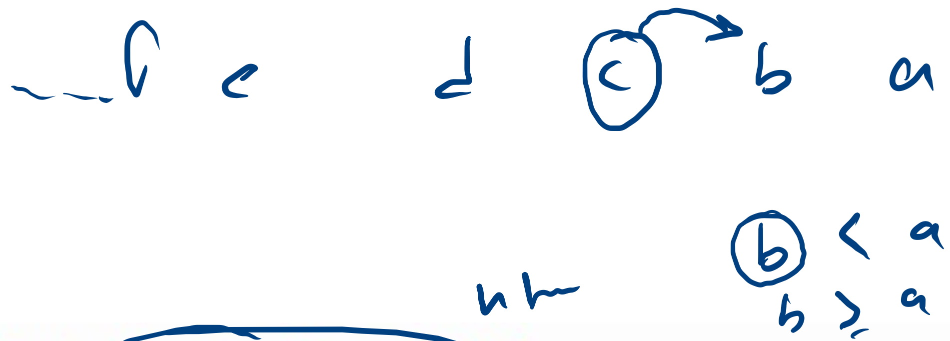


why 2



require check

9x4 ←

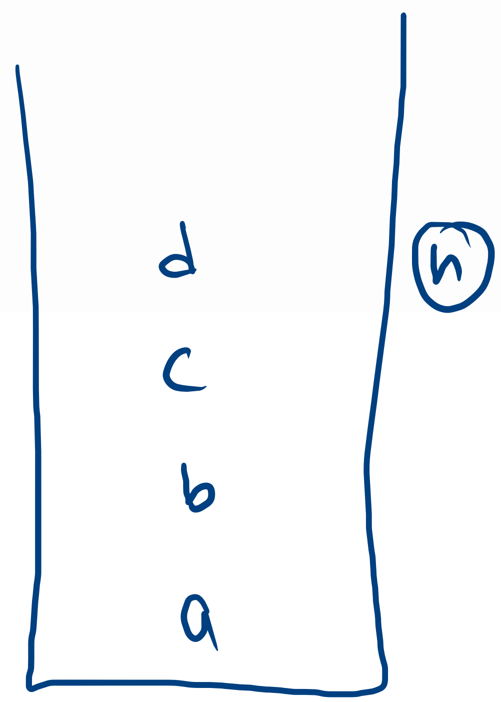


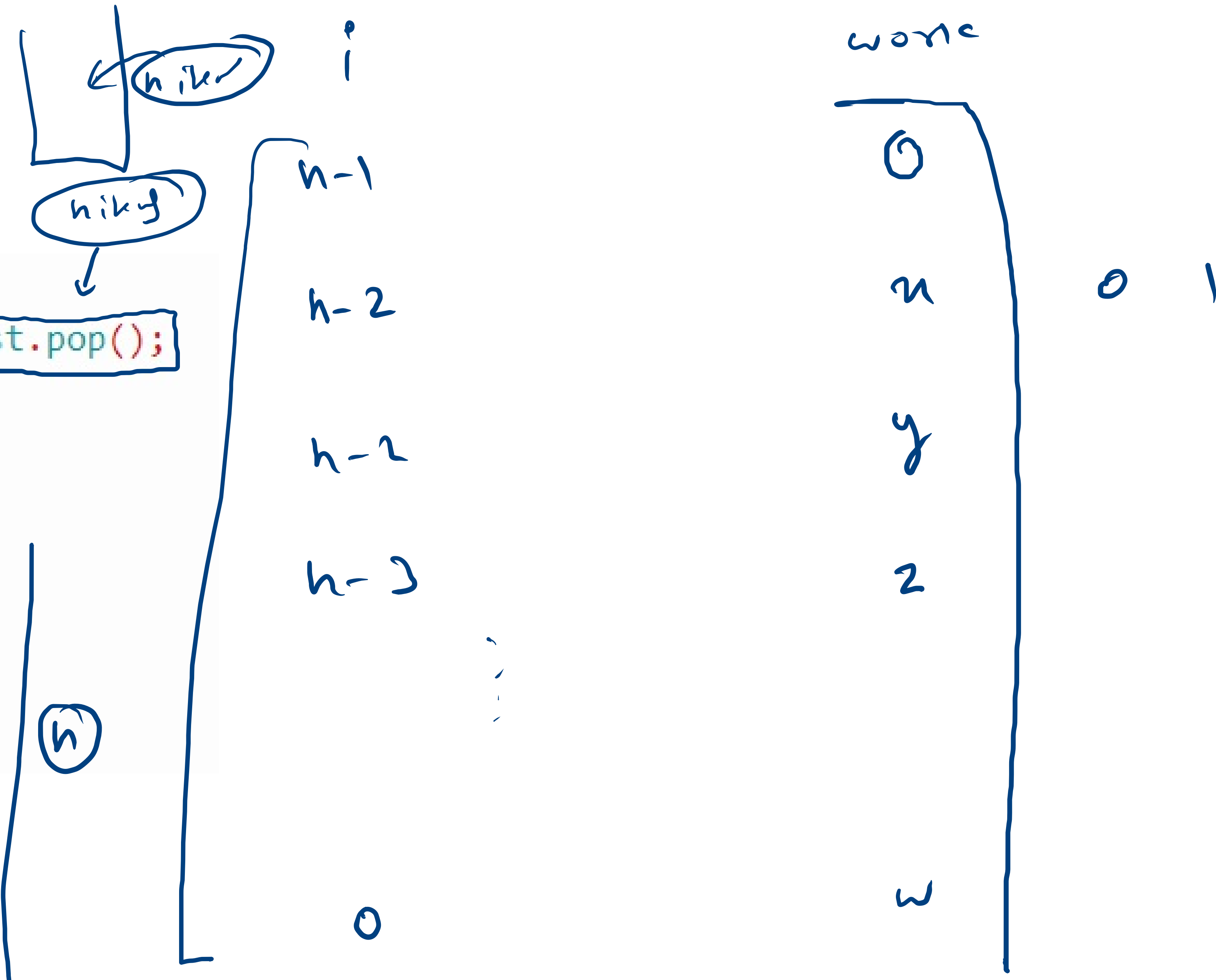
```
for(int i=n-1;i>=0;i--){
    while(st.size() > 0 && st.peek() <= arr[i]) st.pop();

    int ans=-1;
    if(st.size() >0){
        ans = st.peek();
    }
    ngr[i] = ans;

    st.push(arr[i]);
}
```

$n+1+2+\ldots + w \approx n$





2 5 9 3 1 12 6 8 7

12 12 12 -1 8 -1 -1

pop [while (peek < arr[i]) size > 0

pop

ans [if (size = 0)
ans -1
else ans peek

push [push(current)

9
12