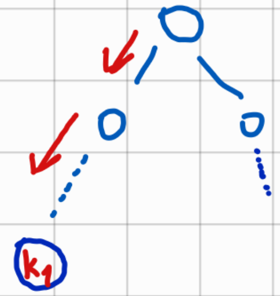
Exercise 6.1

a) k_1 is always the left most node in AVL.
Starting from the root we traverse through left children as far as we can go.



b) We delete the left most node $(i-1)$-times.
The left most node is k_i.

c) $s_l(v)$ decribes the number of nodes in the left subtree of v.

$s_r(v)$ describes the number of nodes in the right subtree of v.

```
v ← Root
rank ← s_l(v) + 1          # holding the current rank of v.
                             rank-1 describes the number of nodes
                             with value less than v.

while ( rank ≠ i )
    if ( i < rank )
        v ← v.leftchild
        rank ← rank - v.rightchild - 1
    else if ( rank > i )
```

```
        v ← rightchild
        rank ← rank + v.leftchild + 1

return v
```

Exercise 6.2

Examples of well formed strings:

ABC
(ABC)
[ABC]
(ABC[])
[()]()
()
[]
(())

Examples of not-well strings: [(])

Use either stack or double-ended queue.

Exercise 6.3

a) Function A(n):

```
        if n ≤ 4
            return n
        else
            return A(n-1) + A(n-3) + 2 A(n-4)
```

b)

$T(1) = T(2) = T(3) = T(4) = 1$

$T(n) = T(n-1) + T(n-3) + T(n-4) + d$

$\Rightarrow T(n) \geq 3\,T(n-4)$

$\Rightarrow \quad T(n) \geq 3T(n-4) \geq 9(T(n-8)) \geq 3^3(T(n-12)) \geq \ldots\ldots \geq 3^k T(n-4k)$

which value of k makes this term 0?

$k = \frac{n}{4}$

Guess $T(n) \geq \frac{1}{3} \cdot 3^{n/4}$ and prove by induction.

c)

memory ← n-dimensional array filled with (-1)s

A_mem(n) :

```
    if memory[n] ≠ -1
        return memory[n]
    if n ≤ 4
        memory[n] ← n
        return n
    else
        A_n ← A_mem(n-1) + A_mem(n-3) + 2·A_mem(n-4)
        memory[n] ← A_n
        return A[n]
```

d)

Linear DP Table sized n.

Dp[k] contains A_k for $1 \leq k \leq n$

Base Cases: DP[1]=1, DP[2]=2, DP[3]=3, DP[4]=4

Calculation of an entry DP[k] with $k \geq 5$: DP[k] = DP[k-1] + DP[k-3] + 2·DP[k-4]

Calculation Order: Compute smallest to largest

DP[n] is the solution

Run Time: $\Theta(n)$, since entry is computed in $\Theta(1)$

Exercise 6.4

$M(k)$: largest position reachable in at most k jumps.

a)

R_n describes the indexes where you can reach in max n steps.

$1 \leq i \leq M[k-1]$ describes the indexes we can reach in max $(n-1)$ steps.

0 steps, 1 step, 1 step, 2 steps, 2 steps, 2 steps, 3 steps, 4 steps, 5 steps, 6 steps

$$A[1 \dots n] = [2, 4, 2, 2, 1, 1, 1, 1, 5, 2]$$

$M[0] = 1$ is reachable

$M[1] = 3$ is reachable

$M[2] = 6$ is reachable

$M[3] = 7$ is reachable

⋮

$M[6]$

$R_1 : 1 \leq i \leq 1$

$R_2 : 1 \leq i \leq 3$

$R_3 : 1 \leq i \leq 6$

⋮

R_6 :

$R_1 = [2+1] = [3]$

$R_2 = [2+1, 4+2, 2+3]$
$= [3, 6, 5]$

$R_3 = [3, 6, 5, 2+4, 1+5, 1+6]$
$= [3, 6, 5, 6, 6, 7]$

⋮

$$\sum_{k=1}^{k} |R_k| = 34$$

b) Only difference is $M'[k-2] \leq M'[k-1]$ instead of $1 \leq M'[k-1]$

0 steps, 1 step, 1 step, 2 steps, 2 steps, 2 steps, 3 steps, 4 steps, 5 steps, 6 steps

$$A[1 \ldots n] = [2, 4, 2, 2, 1, 1, 1, 1, 5, 2]$$

$M[0] = 1$ is reachable

$M[1] = 1 + A[1] = 3$ is reachable

$R_2 : M[0] < i \leq M[1]$

$1 < i \leq 3$

$R_2 = [4+2, 2+3]$
$= [6, 5]$

$M[2] = 6$ is reachable

$M[3] = 7$ is reachable

$\vdots$

$M[6]$

$R_3 : M[1] < i \leq M[2]$

$3 < i \leq 6$

$R_3 = [2+4, 1+5, 1+6]$
$= [6, 6, 7]$

$\vdots$

$R_6 :$

$$\sum_{k=1}^{k} |R_k| = 8 = (\text{length of the array}) - 2$$

first and last index