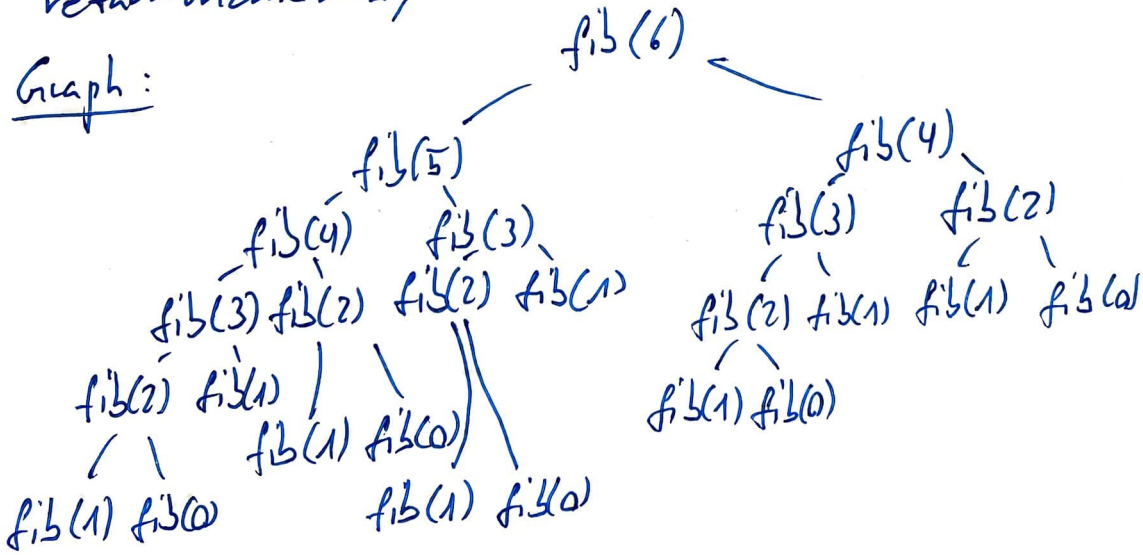
11-105-867 12-122-438
DA Serie 9 D. Marti J. Studer

1.) Wir wählen eine obere Schranke für die Relation.
Wähle $k2^n - b$. Also $T(n) \leq k2^n - b$. Für $n = 0$ gilt
$T(0) = 1 \leq k2^n - b$.
Annahme: Die Relation gilt für $n-1$.
Schritt: $n-1 \mapsto n$
$T(n) = 1 + \sum_{j=0}^{n-1} T(j) \leq 1 + k2^n - b \leq k2^n - b$ für $b \geq 1$.
Also $T(n) = 2^n$.

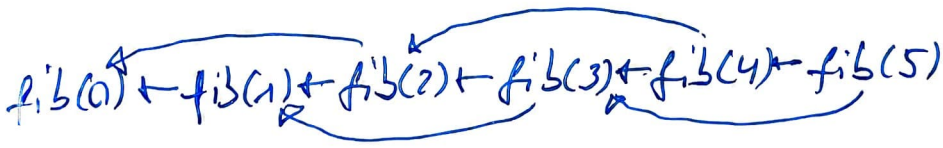
2.) Pseudocode:

```
def fib(n):
  if n==0:
    return 0;
  if n==1:
    return 1;
  if memo[n]!= -1:
    return memo[n];
  memo[n] = fib(n-1) + fib(n-2)
  return memo[n];
```

Graph:



DAG:



3.) Mit dem Algorithmus

```
def LCS (Seq1, Seq2, a, b):
    if ((a == -1) OR (b == -1))
        return 0;
    else if (Seq1[a] == Seq2[b])
        return 1 + LCS(Seq1, Seq2, a-1, b-1)
    else
        return max(LCS(Seq1, Seq2, a-1, b-1),
                   LCS(Seq1, Seq2, a, b-1))
```

erhalten wir 10110

4.) Pseudocode:

a.)

```
def ~~shortest~~ safest Path(T, risk)
    if T(x,y) = T(ENDNODE)
        break;
    else if T(x+1, y) > T(x, y+1)
        risk += T(x,y).risk
        safestPath(T(x+1, y), risk)
    else if T(x+1, y) < T(x, y+1)
        risk += T(x, y).risk
        safestPath(T(x, y+1), risk)
    end
end
```

b.) Der Algorithmus startet bei T und vergleicht das Risiko im Kasten daneben mit dem darunter. Der Algorithmus (Taxifahrer) wählt dann immer die günstigere Route von beiden. Befindet sich der Taxifahrer am Rand, hat er nur eine mögliche Route, die er nehmen kann.

In der Tabelle sieht die Route dann folgendermaßen aus.

~~1~~	1	9	2	~~3~~	5	7	
8	~~5~~	3.	15	2	7	9	
7	10	4	15	5	6	6	
12	3	5	9	15	14	7	
10	2	~~3~~	~~1~~	~~7~~	~~4~~	~~5~~	