Dynamisches Programmieren

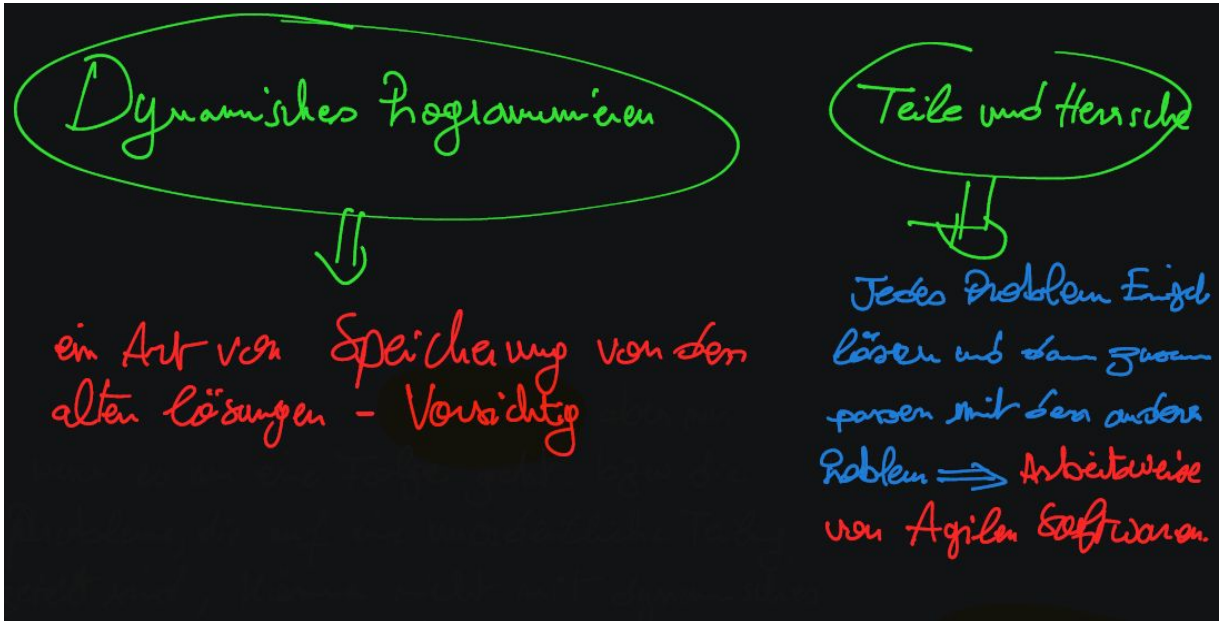
Anass Halime,
George Antoun,
Kawa Ramadan

Gliederung:

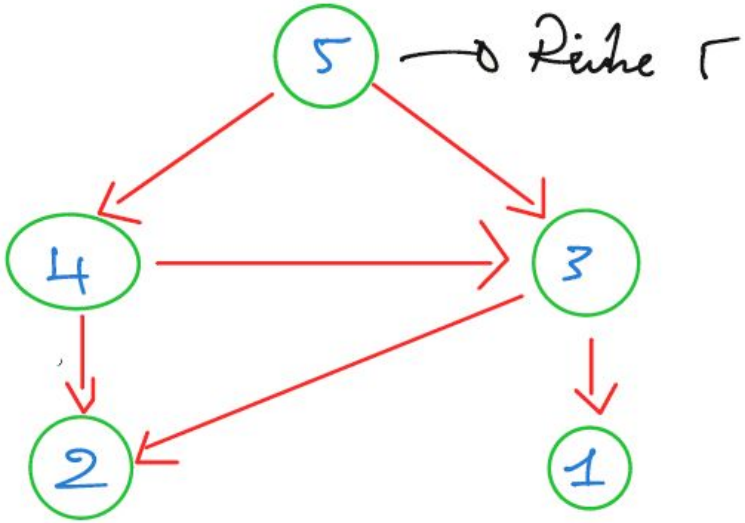
1. Definitionen - ”Hallo Welt”
2. Fibonacci _Zahlen
3. Checkpoint _ Zusammenfassung
4. Das 0/1 Rucksackproblem
5. Wegfliegen von Theorie → Praxisteil (Java Code)
6. Rundreiseproblem
7. Bellman-Ford-Algorithmus
8. Geldwechselproblem.
9. Conclusion

Checkpoint

- Unterschied zwischen “teile und herrsche ” und “dynamisches Programmieren”.



Beispiel:

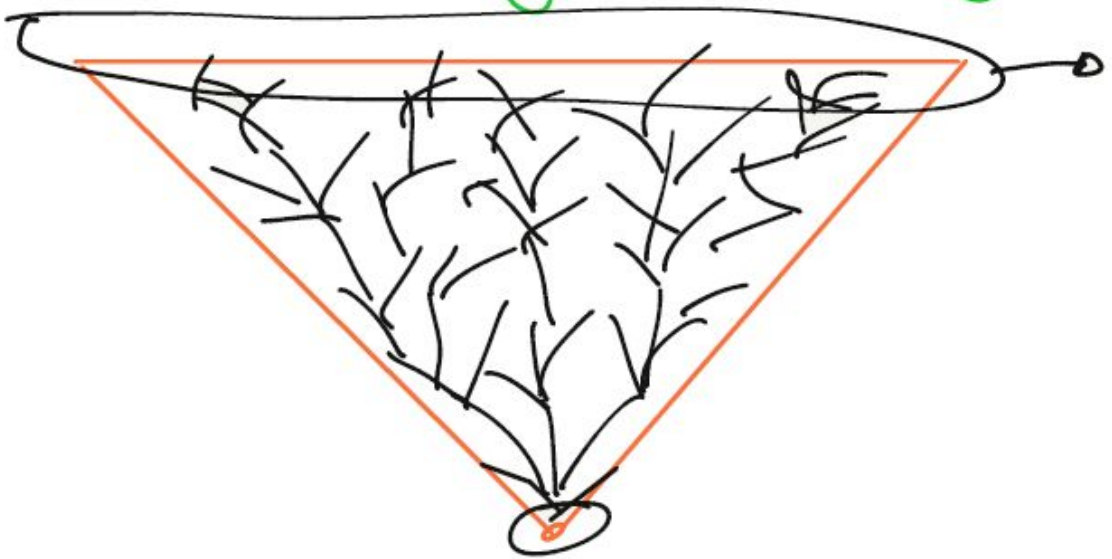


- CPU - Überlastung kürzen.

Fib(7)=

Fib(13) = 0, 1, 1, 2, 3, 5, 8, 13

Das kann ich wie ein umgedrehte Pyramide.



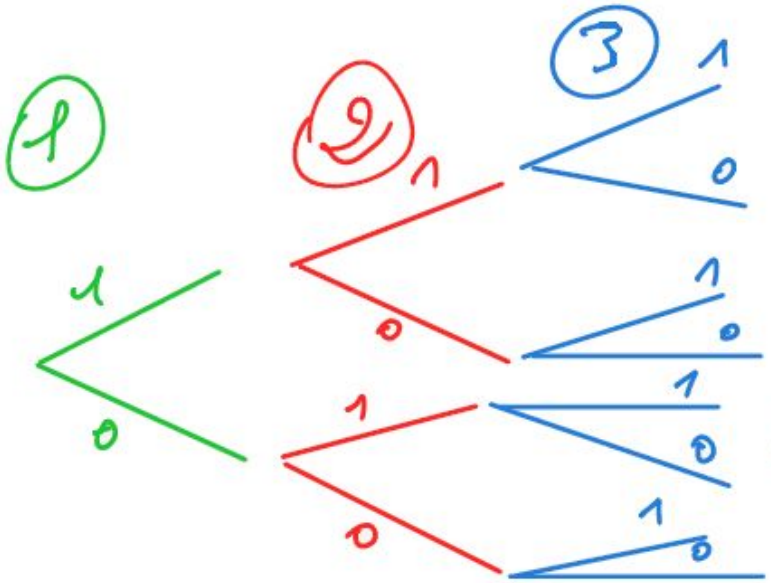
das 0/1 Rücksack-Problem

Geschenk (i)	1	2	3	4
Wert	15 €	9 €	6 €	2 €
Gewicht	8 kg	3 kg	4 kg	3 kg

Ziel:

das ziel ist, den Gesamtwert von den Geschenken zu maximieren ohne die maximale die Kapazität (Gewicht) zu überschreiten.

Beispiel : Lösung



in diesem Fall werde ich.
am Ende insgesamt 2^3 Fälle.

Skizze 2

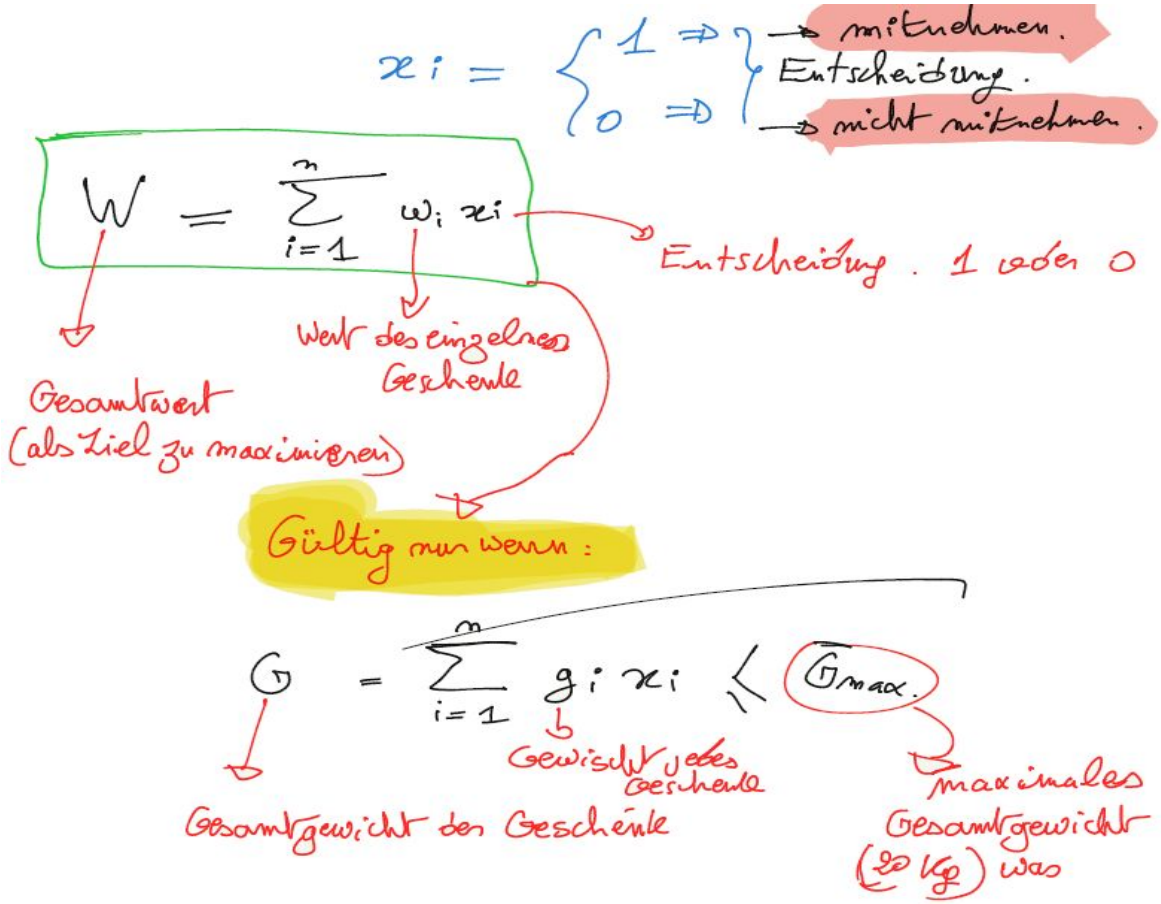
noch ein problem !

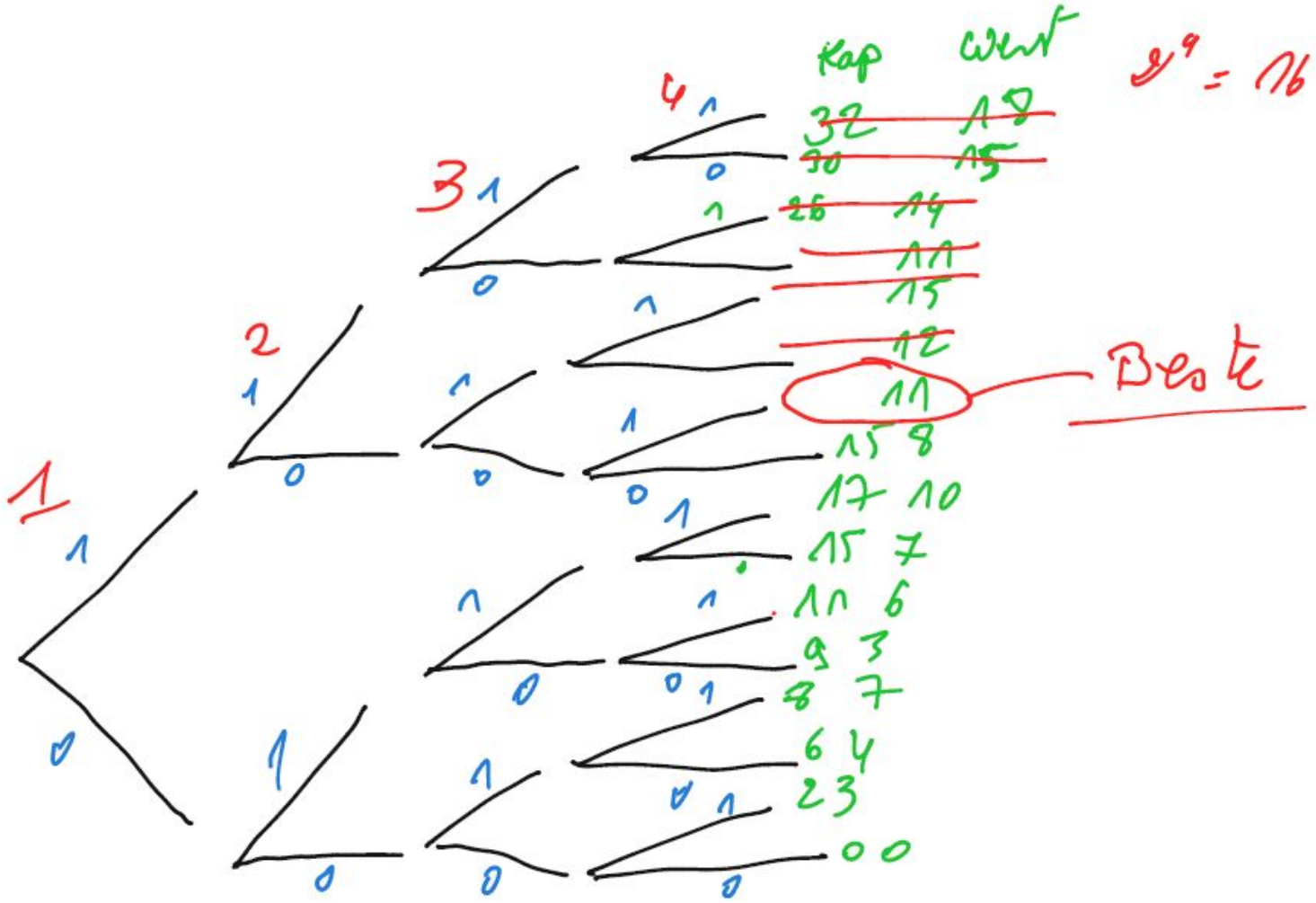
Problem:

bei n Gegenständen wird CPU 2^n Fall bearbeiten müssen

→ Überlastung & Zeitaufwand.

Mathematische Formel :




Кар
Вес
$g^9 = 16$
Best

Lösung:

1. Rekursive Solution
2. Memorize intermediate Results
3. Bottom - up

Rundreise-Problem

- Geschichte dieses Problem.
- *Traveling salsmen problem* → NP. (Non- deterministic Polynomial)
- 868331761881188649551819440128000000 0 Fälle für 33 Städte.
- 33! = 868331761881188649551819440128000000 ca. → 10^{37}