Search.02

1) Suchverfahren

Gesucht ist ein Pfad von **Würzburg (Wü)** nach **München (Mü)**. Wir verwenden *Graph-Search* mit „keine Zyklen“ (besuchte Knoten werden nicht erneut expandiert) und Tie-Break *alphabetisch*. Für DFS werden Nachbarn in *umgekehrter Alphabet-Reihenfolge* auf den Stack gelegt, damit beim Pop die alphabetisch erste Stadt als Nächstes dran ist.

1a) Tiefensuche (DFS) — Datenstruktur: Stack

Schritt	Stack (Top rechts)	Bemerkung
1	[Wü]	Start
2	[WüEr, WüFr, WüNü]	Wü expandiert; Nachbarn gepusht (rev. alph.)
3	[WüEr, WüFr, WüNüSt, **WüNüMü**]	WüNü expandiert; erzeugt Mü und St
4	[WüEr, WüFr, WüNüSt]	**Pop WüNüMü** = Zielpfad erreicht

Ergebnis: Pfad Wü → Nürnberg → München, Kosten $103 + 167 =$ **270 km**; Durchläufe: **4**; max. Stackgröße: **4**.

1b) Breitensuche (BFS) — Datenstruktur: Queue

Schritt	Queue (links = Nächstes)	Bemerkung
1	[Wü]	Start
2	[WüEr, WüFr, WüNü]	Wü expandiert; 3 Nachfolger
3	[WüFr, WüNü]	Erfurt expandiert; keine neuen
4	[WüNü, WüFrKas, WüFrMa]	Frankfurt expandiert; Kassel, Mannheim
5	[WüFrKas, WüFrMa, **WüNüMü**, WüNüSt]	Nürnberg expandiert; Mü, St
6	[WüFrMa, **WüNüMü**, WüNüSt]	Kassel expandiert
7	[**WüNüMü**, WüNüSt]	Zielpfad wird dequeued

Ergebnis: Pfad Wü → Nürnberg → München, Kosten **270 km**; Durchläufe: **7**; max. Queuegröße: **4**.

1c) A*-Suche (mit *ursprünglicher* Heuristik) — Datenstruktur: sortierte Queue

Die im Blatt gegebene Heuristik ist *nicht zulässig* (z. B. $h(\text{Nürnberg}) = 537\,\text{km} > h^{(\text{Nürnberg})=167}\,\text{km}$). Ein typischer A*-Ablauf ergibt:

Schritt	Prioritätsqueue (kleinstes $f = g{+}h$ zuerst)	Bemerkung
1	[Wü 0+170=170]	Start
2	[WüFr 217+100=317, WüEr 186+400=586, WüNü 103+537=640]	Wü expandiert
3	[WüFrMa 302+200=502, WüFrKas 390+460=850, WüEr 586, WüNü 640]	Fr expandiert
4	[WüFrMaKar 382+10=392, WüFrKas 850, WüEr 586, WüNü 640]	Ma expandiert
5	[WüFrMaKarAu 632+0=632, WüFrKas 850, WüNü 640]	Kar expandiert; Au entsteht
6	[**WüFrMaKarAuMü** 716+0=716, WüFrKas 850, WüNü 640]	Ziel vorhanden ⇒ Ende

Ergebnis: Pfad Wü → Fr → Ma → Kar → Au → Mü, Kosten $217 + 85 + 80 + 250 + 84 =$ **716 km** (suboptimal; Grund: unzulässige Heuristik).

1d) Vergleich

Algorithmus	Durchläufe (Expansionen)	max. Speicher	Pfadkosten
Tiefensuche (DFS)	4	4	270 km
Breitensuche (BFS)	7	4	270 km
A* (Heuristik alt)	8	4	716 km

Bemerkung. BFS liefert hier den optimalen Weg; A* nur dann, wenn die Heuristik zulässig ist.

2) Heuristik

2a) Diskussion der Restkostenabschätzung

Für A* (Tree-Search) muss die Heuristik h *zulässig* sein:

$$h(n) \le h^{(n)} \quad \text{für alle } n, \qquad h(\text{Ziel})=0.$$

Im Blatt ist $h(\text{Nürnberg}) = 537\,\text{km}$, während die echte minimale Restdistanz $h^{(\text{Nürnberg})=167}\,\text{km}$ (direkte Kante Nürnberg–München) beträgt. ⇒ **nicht zulässig**.

Minimale Korrektur: Wir ändern nur Nürnberg und setzen

$$\boxed{h(\text{Nürnberg}) = 160\ \text{km}} \quad (\le 167\ \text{km}).$$

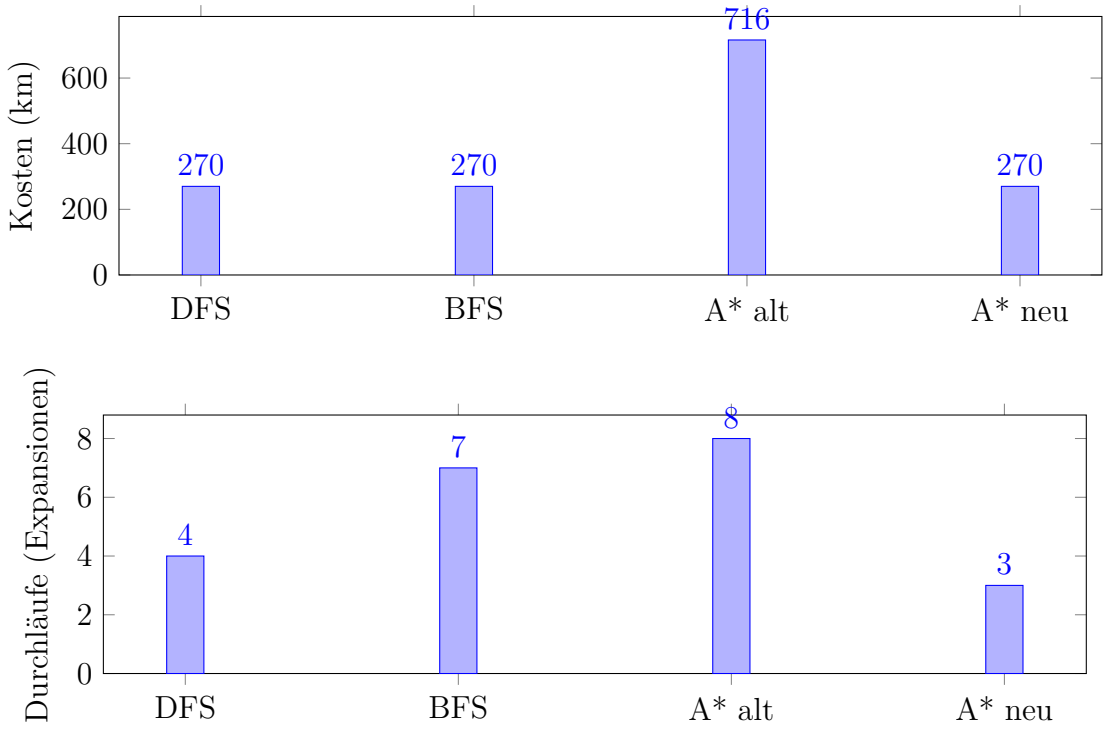
2b) A* mit *neuer* Heuristik (zulässig)

Mit $h(\text{Nürnberg}) = 160\,\text{km}$ wird A* sofort optimal:

Expandiert	$g(n)$	$h(n)$	$f(n)$
Würzburg	0	170	170
Nürnberg	103	160	263
München	270	0	270

Ergebnis: Wü → Nürnberg → München, Kosten **270 km**; nur **3 Expansionen**.

Vergleich (Kosten & Expansionen).



Fazit. Mit zulässiger Heuristik liefert A* in diesem Graphen sofort den optimalen Pfad (270 km) und benötigt dabei die wenigsten Expansionen. DFS kann je nach Reihenfolge schnell oder sehr langsam sein; BFS ist robust und findet stets den optimalen Weg hinsichtlich Tiefe, benötigt hier aber mehr Expansionen.