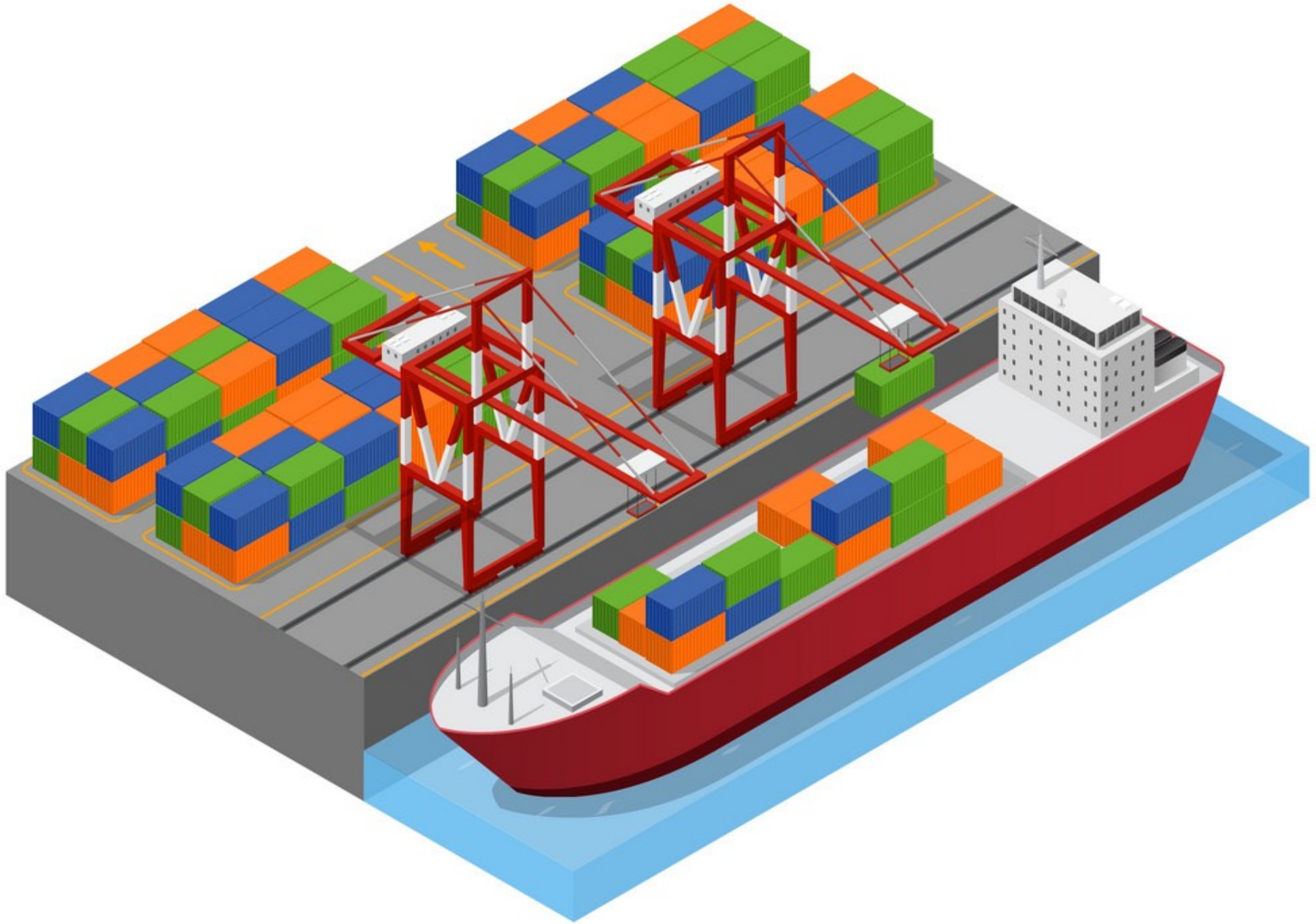
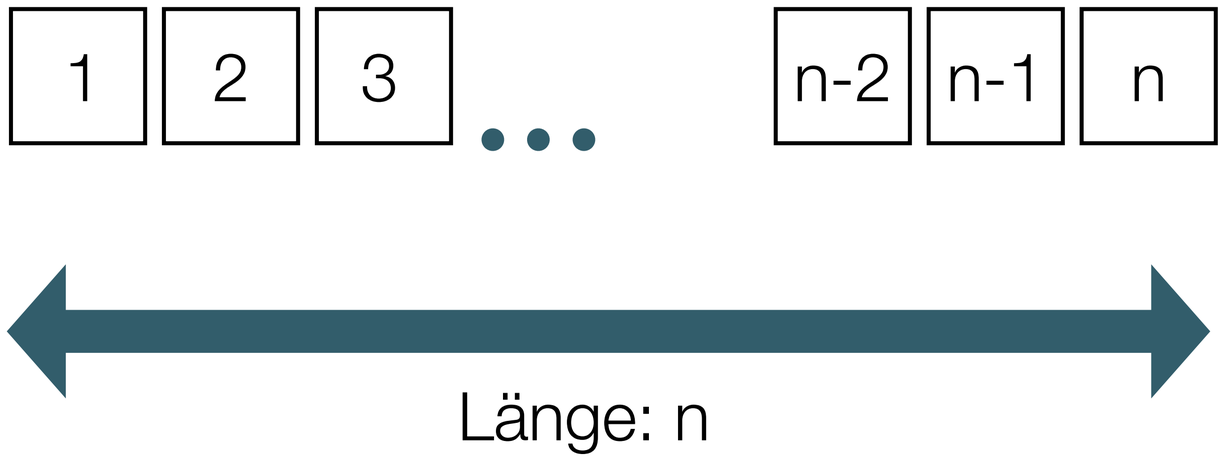
G: Safe Harbour

Von Tobias Gloger und Magnus Bühler



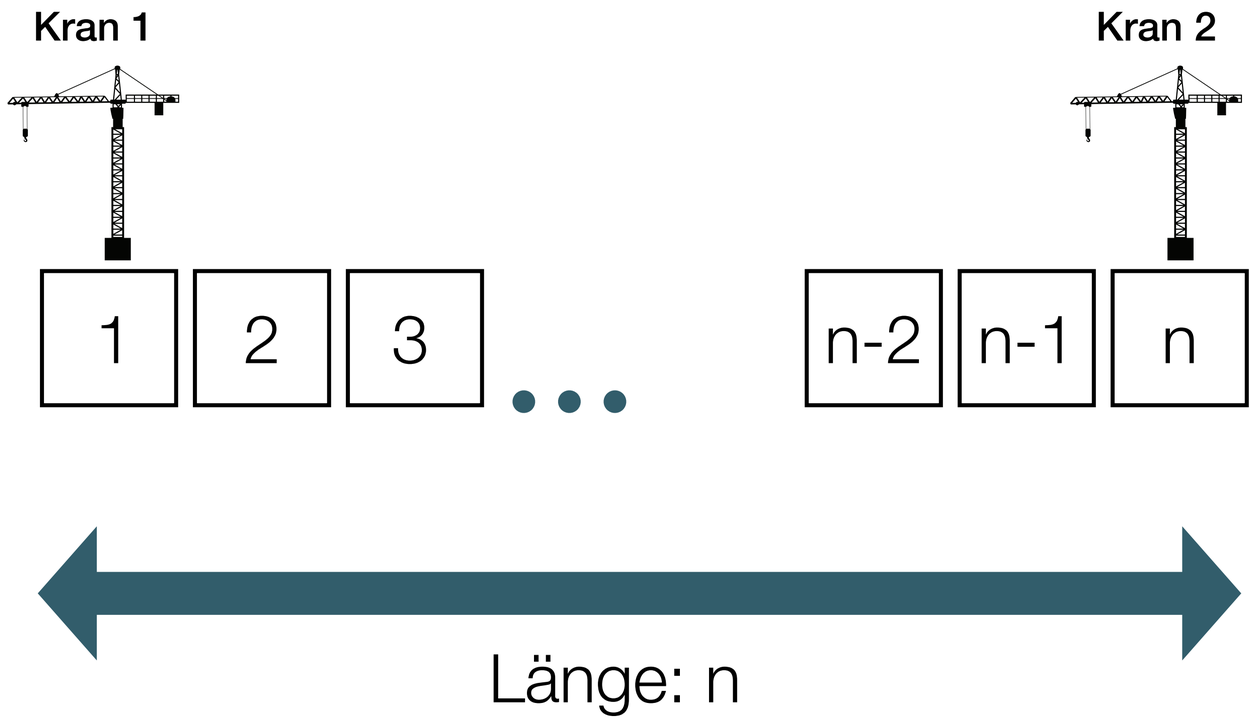
Problembeschreibung

- Länge des Hafens: n
- Durchnummeriert von 1 bis n



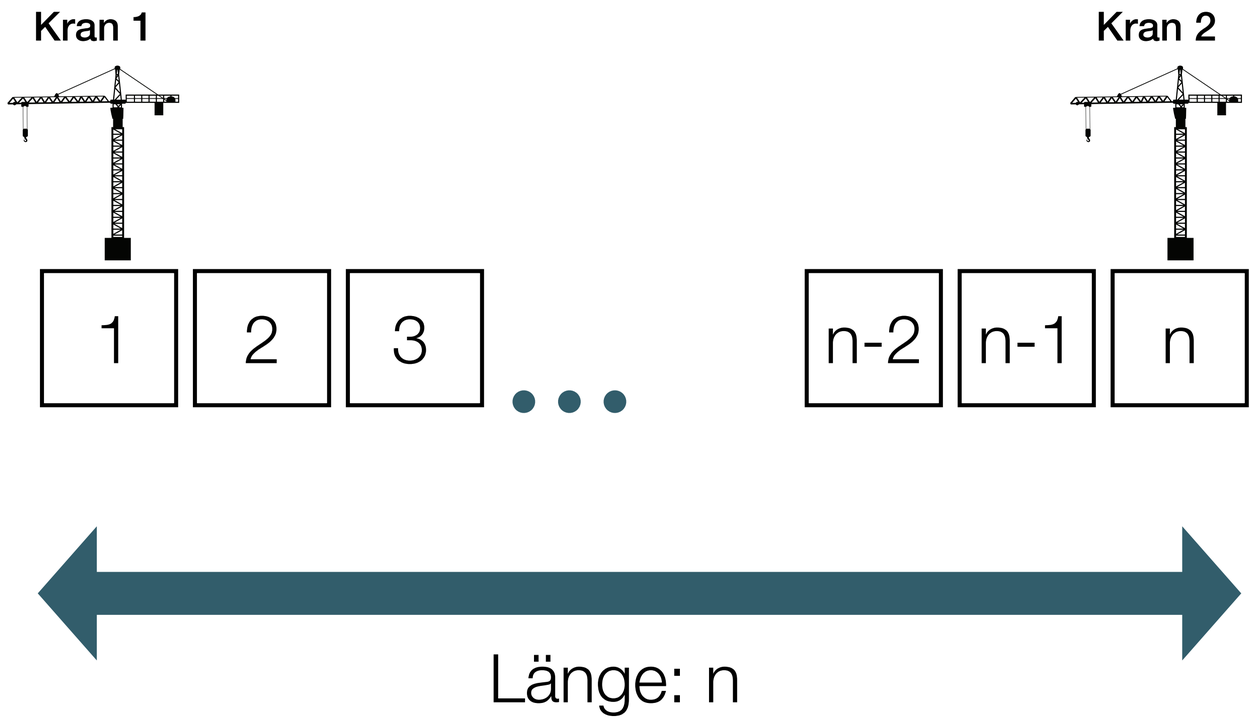
Problembeschreibung

- Länge des Hafens: n
- Durchnummeriert von 1 bis n
- Start- & Endkonfiguration:
  - Kran 1 steht auf der 1
  - Kran 2 steht auf n



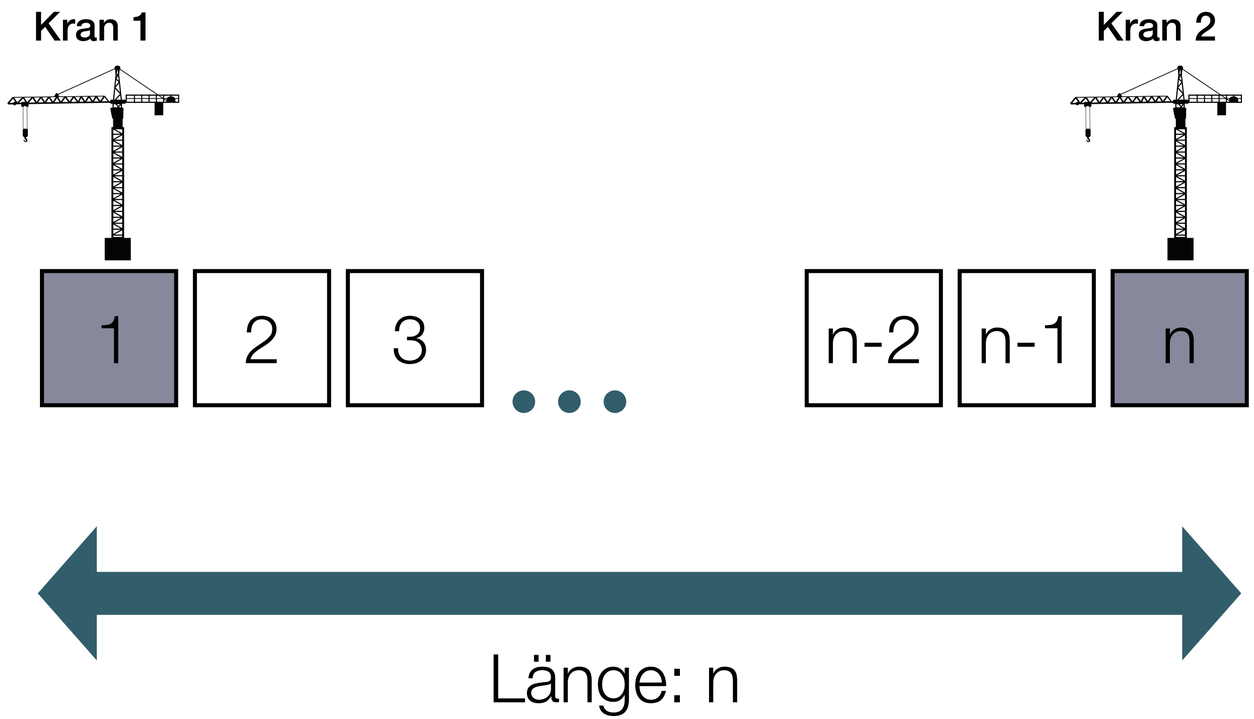
Problembeschreibung

- Länge des Hafens: n
- Durchnummeriert von 1 bis n
- Start- & Endkonfiguration:
  - Kran 1 steht auf der 1
  - Kran 2 steht auf n
- Beide Kräne bekommen geordnete Aufgaben (= Hafenpositionen), die sie abarbeiten müssen

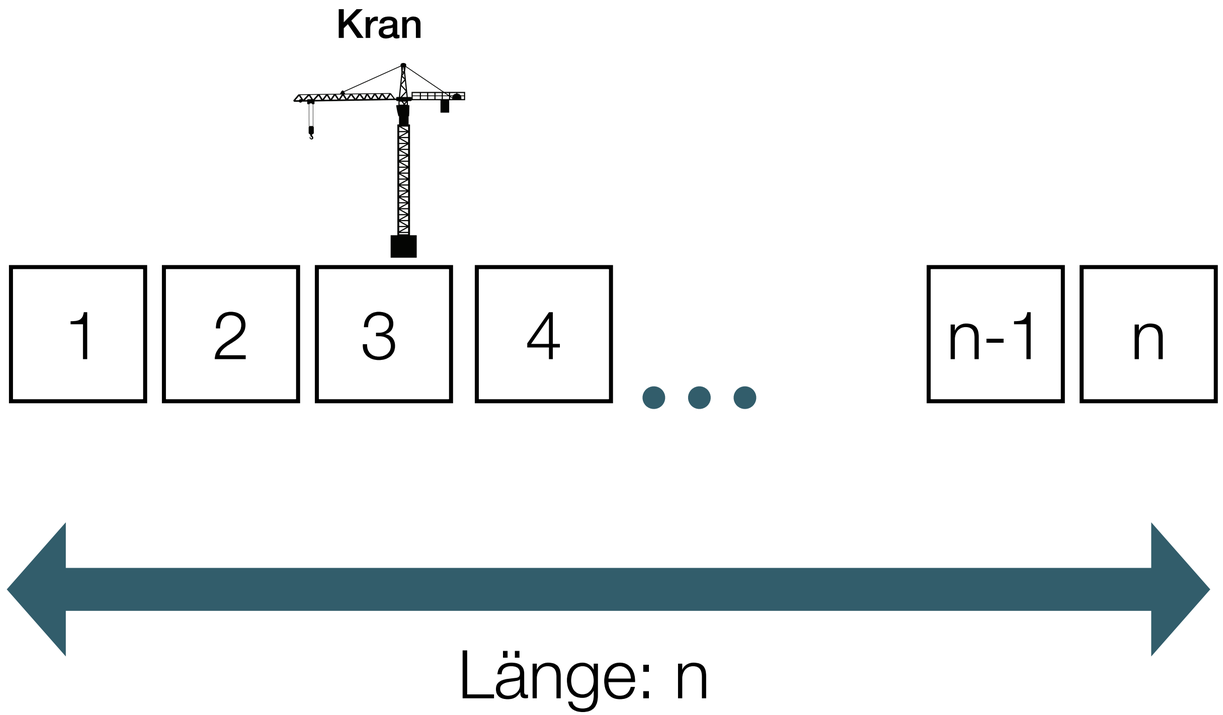


Problembeschreibung

- Länge des Hafens: n
- Durchnummeriert von 1 bis n
- Start- & Endkonfiguration:
  - Kran 1 steht auf der 1
  - Kran 2 steht auf n
- Die erste & letzte Aufgabe von Kran 1 ist 1
- Die erste & letzte Aufgabe von Kran 2 ist n

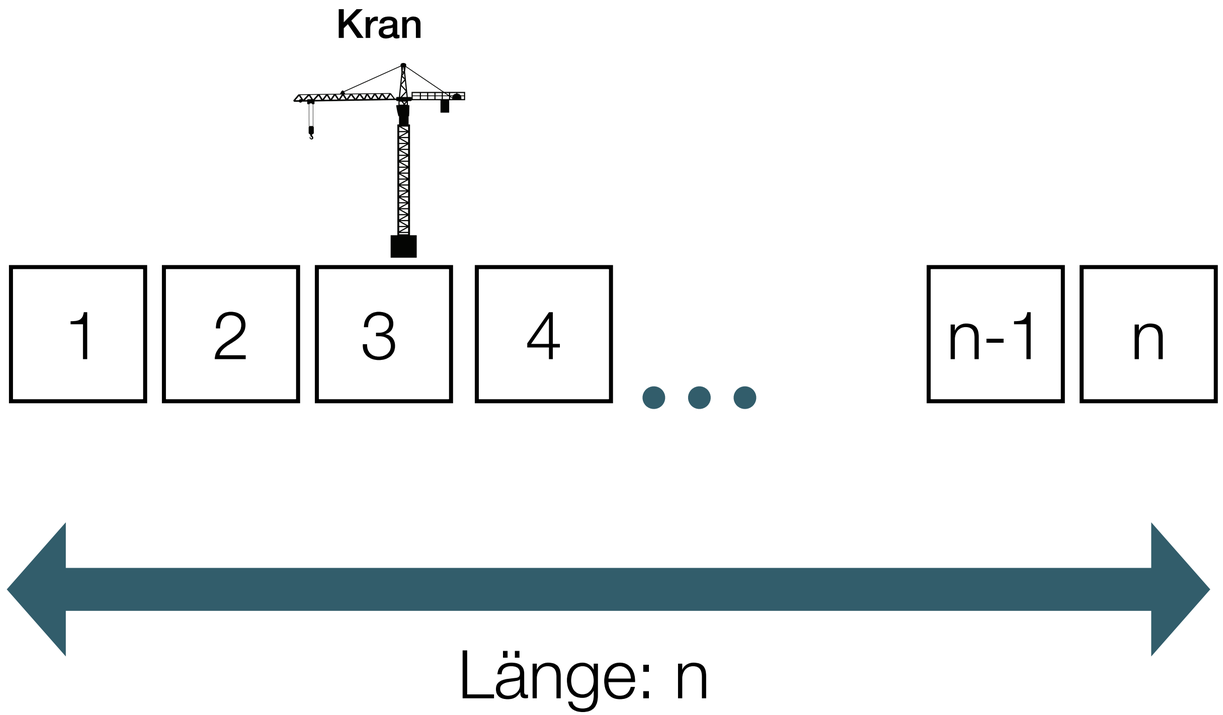


Problembeschreibung: Mögliche Aktionen in einem Zeitschritt



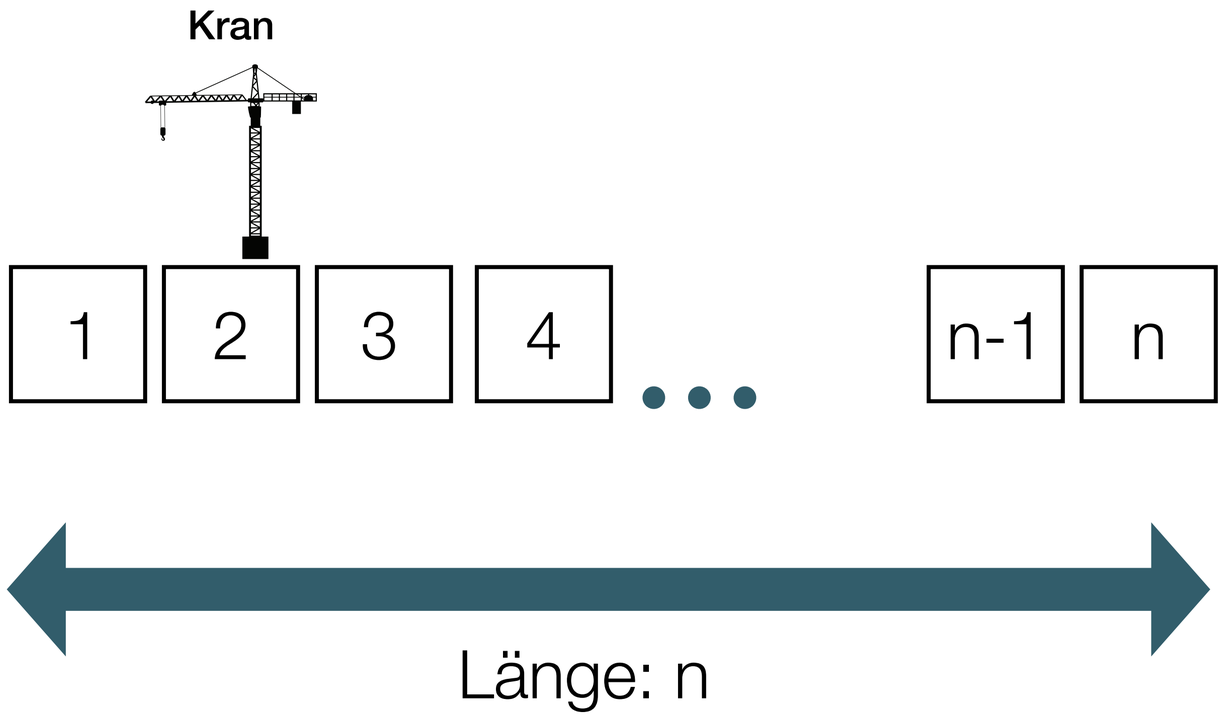
Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

- Schritt nach links



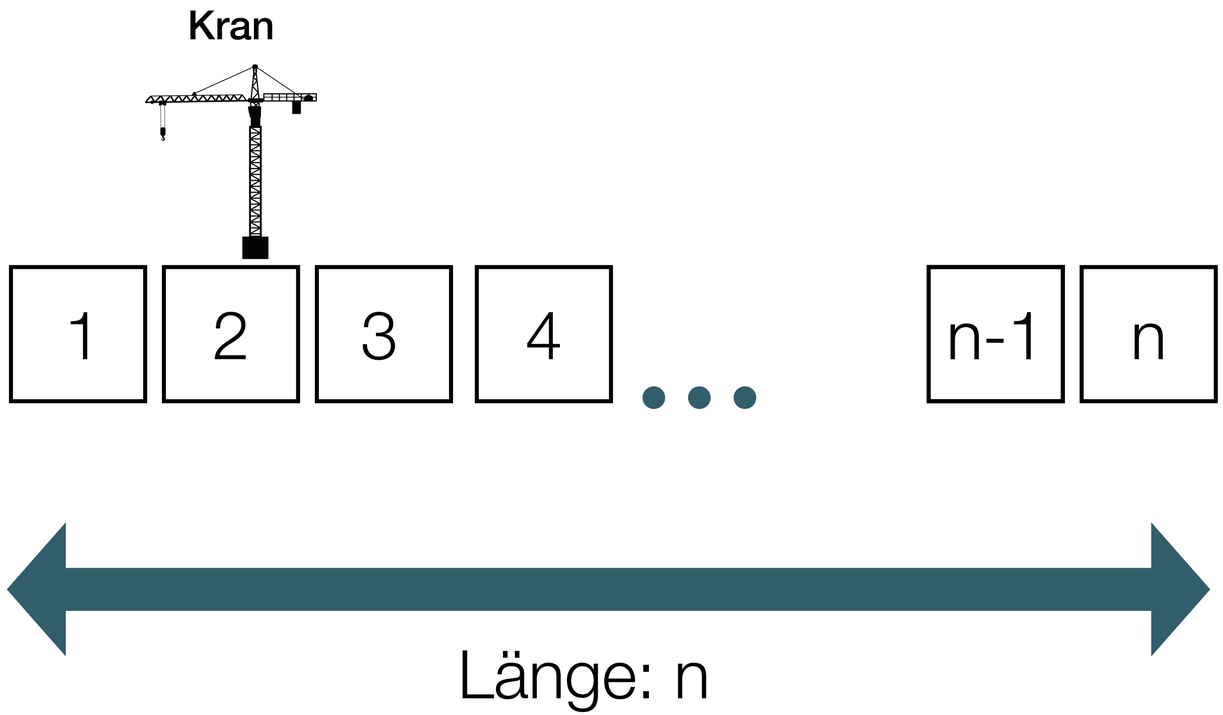
Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

- Schritt nach links



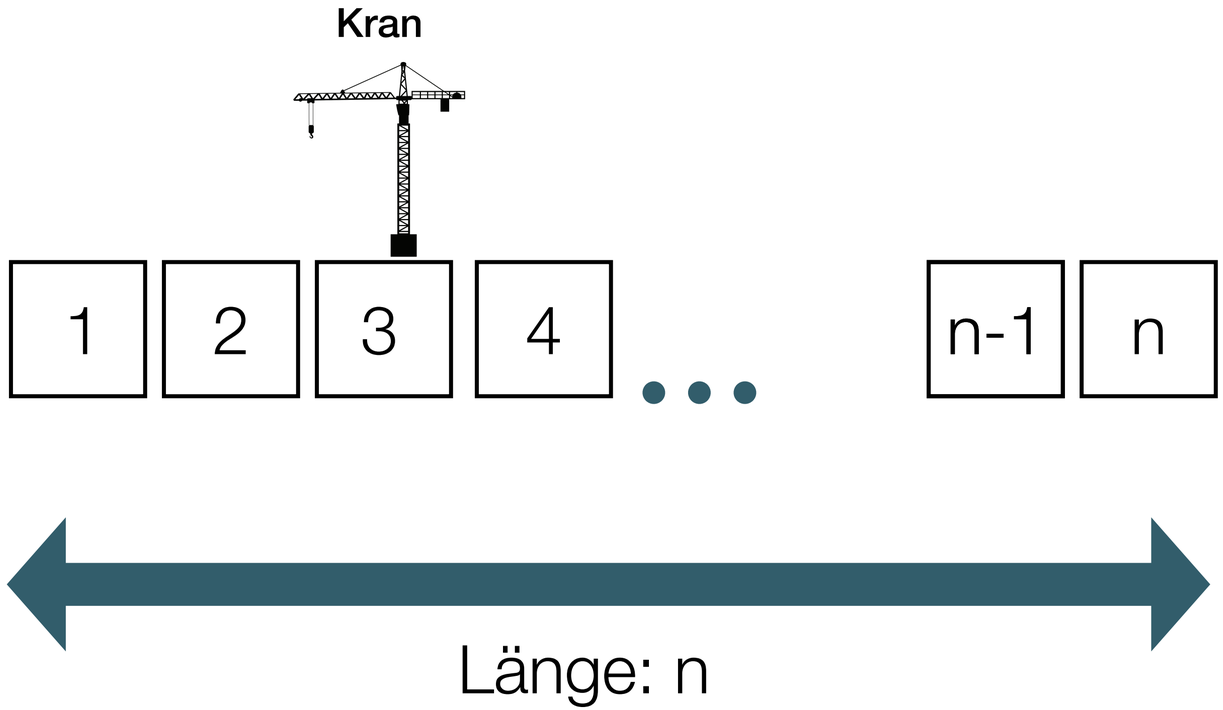
Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

- Schritt nach links
- Schritt nach rechts



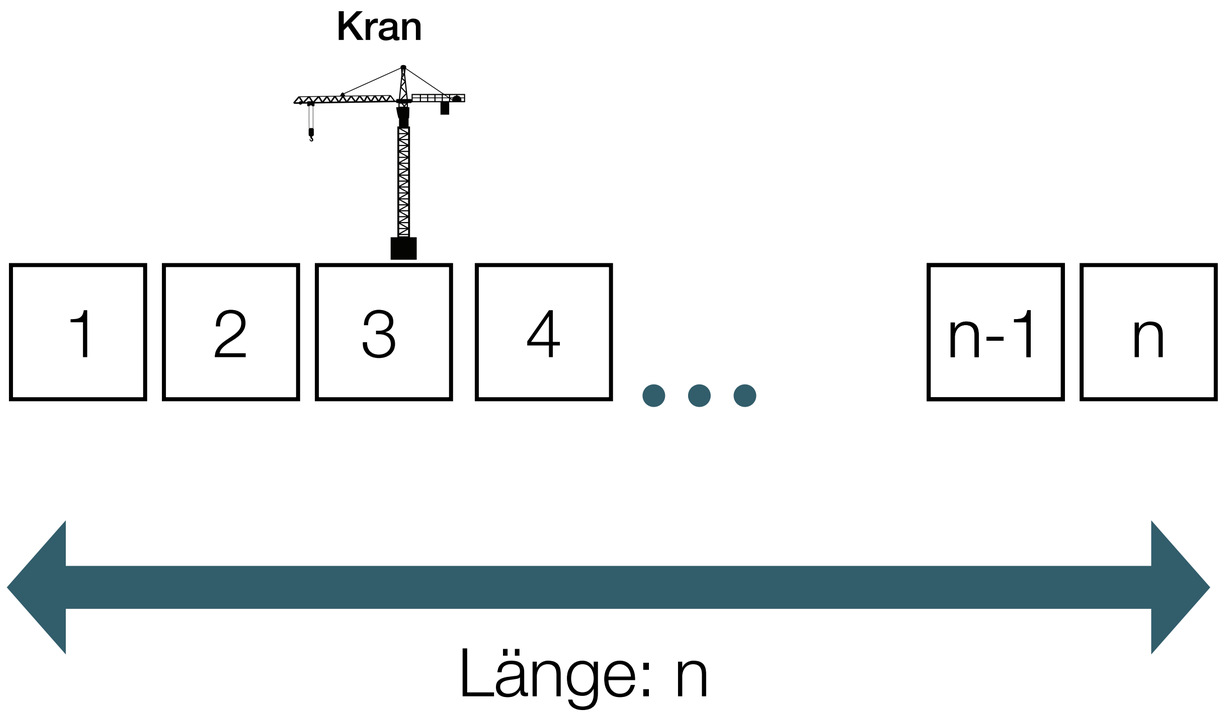
Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

- Schritt nach links
- Schritt nach rechts



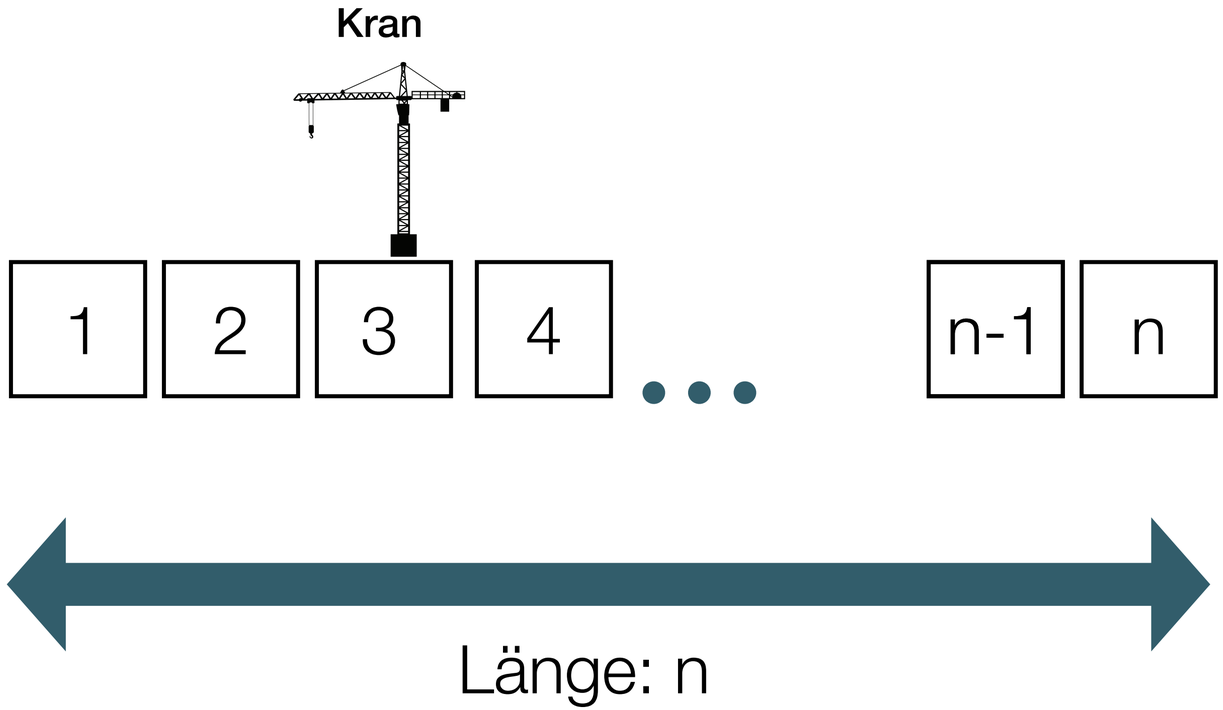
Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

- Schritt nach links
- Schritt nach rechts
- Warten



Problembeschreibung: Mögliche Aktionen in einem Zeitschritt

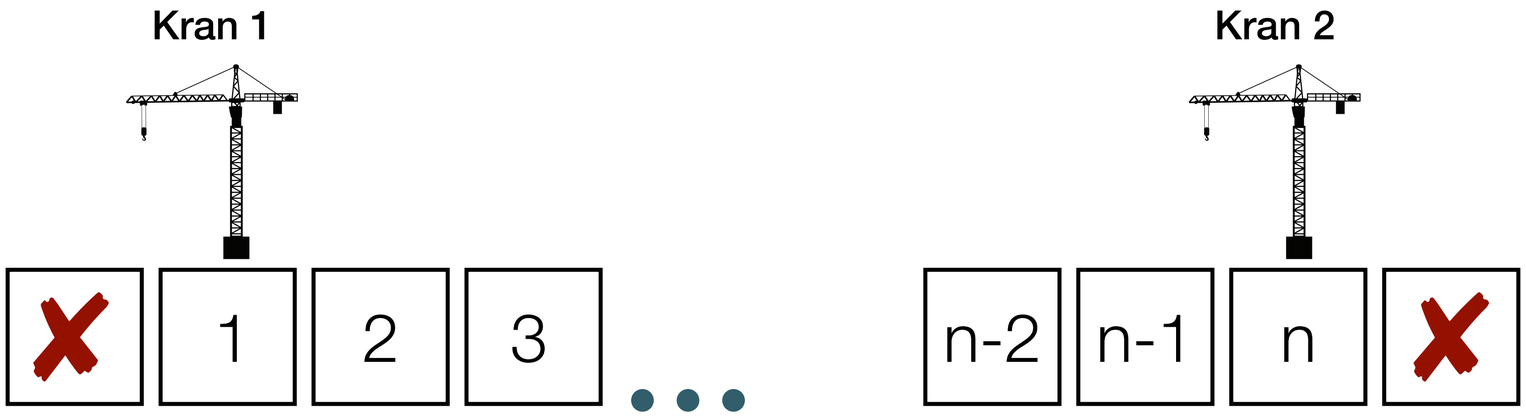
- Schritt nach links
- Schritt nach rechts
- Warten
- Auf- & abladen (= Aufgabe abarbeiten)



Problembeschreibung: Einzuhaltende Beschränkungen

- Kräne müssen immer im Hafenbereich sein

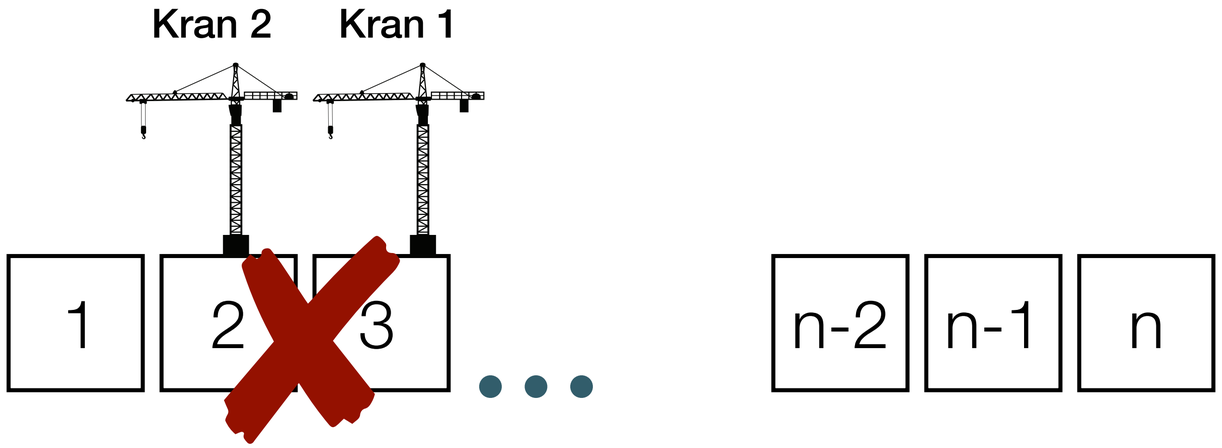
$1 \leq \text{Position} \leq n$



Problembeschreibung: Einzuhaltende Beschränkungen

- Kräne müssen immer im Hafenbereich sein $1 \leq \text{Position} \leq n$

- Kran 1 muss immer links von Kran 2 bleiben $\text{Kran 1} < \text{Kran 2}$

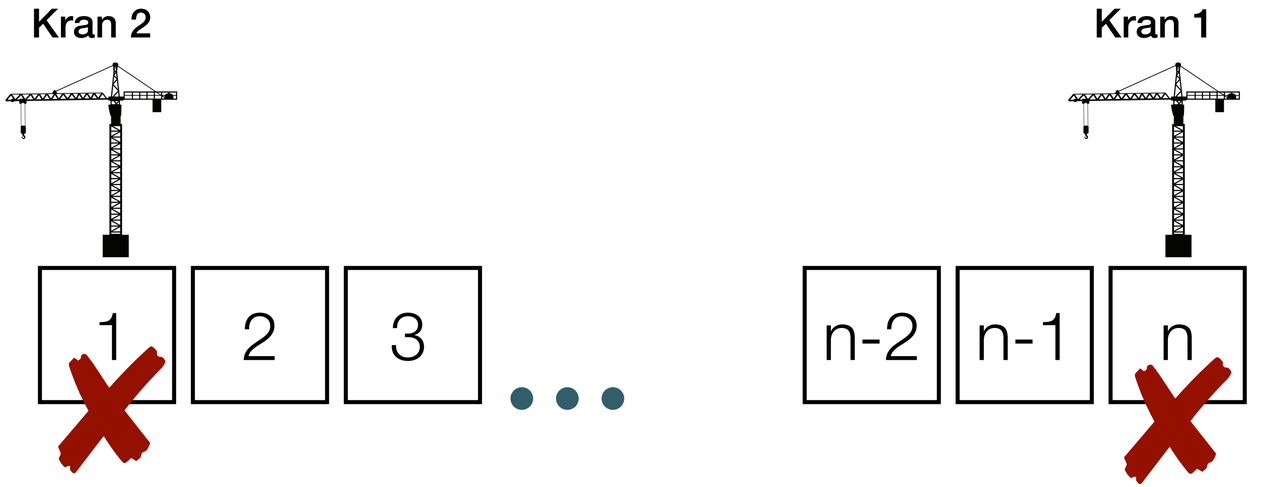


Problembeschreibung: Einzuhaltende Beschränkungen

- Kräne müssen immer im Hafenbereich sein $1 \leq \text{Position} \leq n$

- Kran 1 muss immer links von Kran 2 bleiben $\text{Kran 1} < \text{Kran 2}$

→ Kran 1:[1,n-1]
Kran 2:[2,n]



Problembeschreibung: Einzuhaltende Beschränkungen

- Kräne müssen immer im Hafenbereich sein — $1 \leq \text{Position} \leq n$
- Kran 1 muss immer links von Kran 2 bleiben — $\text{Kran 1} < \text{Kran 2}$
- Die Reihenfolge der Aufgaben für Kran 1 und Kran 2 sind vorgegeben

Gesucht:

Wie viele Zeitschritte benötigen die beiden Hafenkräne mindestens,

um ihre Aufgaben zu erledigen?

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:

Beispieleingabe

```
3 2 4
```

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:
  - $2 \leq n \leq 2000$ die Länge des Hafens angibt

Beispieleingabe

```
3 2 4
```

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:
  - $2 \leq n \leq 2000$ die Länge des Hafens angibt
  - $2 \leq a \leq 50$ die Anzahl an Aufgaben für Kran 1 angibt

Beispieleingabe

```
3 2 4
```

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:
  - $2 \leq n \leq 2000$ die Länge des Hafens angibt
  - $2 \leq a \leq 50$ die Anzahl an Aufgaben für Kran 1 angibt
  - $2 \leq b \leq 50$ die Anzahl an Aufgaben für Kran 2 angibt

Beispieleingabe

```
3 2 4
```

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:
  - $2 \leq n \leq 2000$ die Länge des Hafens angibt
  - $2 \leq a \leq 50$ die Anzahl an Aufgaben für Kran 1 angibt
  - $2 \leq b \leq 50$ die Anzahl an Aufgaben für Kran 2 angibt
- Zweite Zeile mit a Integer $k_1, \ldots, k_a$ mit $(\forall i: 1 \leq k_i \leq n-1)$ stellt die Aufgabenliste von Kran 1 dar

Beispieleingabe

```
3 2 4
1 1
```

Eingabe

- Erste Zeile drei Integer n, a, b, wobei:
  - $2 \leq n \leq 2000$ die Länge des Hafens angibt
  - $2 \leq a \leq 50$ die Anzahl an Aufgaben für Kran 1 angibt
  - $2 \leq b \leq 50$ die Anzahl an Aufgaben für Kran 2 angibt
- Zweite Zeile mit a Integer $k_1, \ldots, k_a$ mit $(\forall i: 1 \leq k_i \leq n-1)$ stellt die Aufgabenliste von Kran 1 dar
- Dritte Zeile mit b Integer $l_1, \ldots, l_b$ mit $(\forall j: 2 \leq l_j \leq n)$ stellt die Aufgabenliste von Kran 2 dar

Beispieleingabe

```
3 2 4
1 1
3 3 2 3
```

Mathematische Modellierung: Notation

- n = Länge des Hafens
- $P_{\text{Kran 1}} = p_1, \ldots, p_a$ ($\forall p: 1 \leq p_i \leq n-1$) = geordnete Aufgabenliste von Kran 1
- $Q_{\text{Kran 2}} = q_1, \ldots, q_b$ ($\forall q: 2 \leq q_j \leq n$) = geordnete Aufgabenliste von Kran 2
- p_i = Position der Aufgabe Nummer i von Kran 1
- q_j = Position der Aufgabe Nummer j von Kran 2

Mathematische Modellierung: Graph

Definiere Graph aus dem Modell

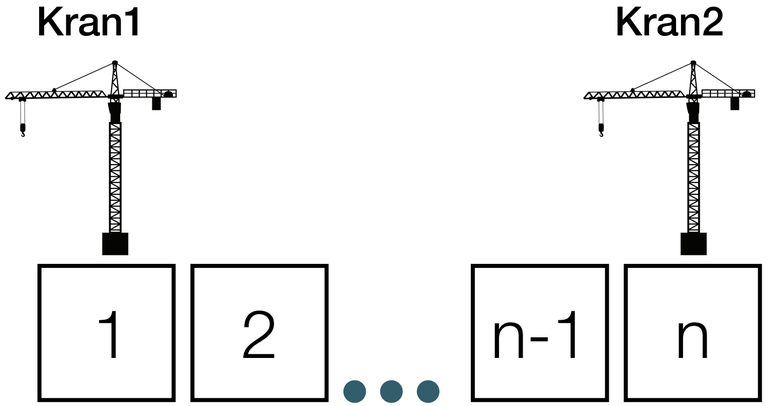
Mathematische Modellierung: Definition

- $f_a(i,j,x) =_{def}$ der frühstmögliche Zeitpunkt, dass:
  - Kran 1 Aufgabe **i** gemacht hat
  - Kran 2 Aufgabe **j** gemacht hat
  - Kran 1 auf Position p_i steht
  - Kran 2 auf Position **x** steht
- $f_b(i,j,x) =_{def}$ analog

$f_a(i,j,x)$

Mathematische Modellierung: Beobachtung 1

- $\mathbf{f_a(1,1,n)} =_{def}$ der frühstmögliche Zeitpunkt, dass:
  - Kran 1 Aufgabe **1** gemacht hat
  - Kran 2 Aufgabe **1** gemacht hat
  - Kran 1 auf Position **1** steht
  - Kran 2 auf Position **n** steht



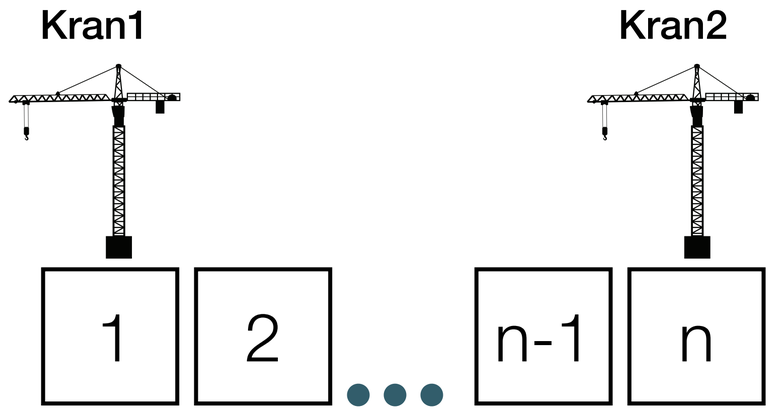
- $\mathbf{f_a(1,1,n)} = 1 = \mathbf{f_b(1,1,1)}$

- „Die optimalen Kosten, dass beide Kräne ihre erste Aufgabe abgearbeitet haben sind immer 1“

Mathematische Modellierung : Beobachtung 2

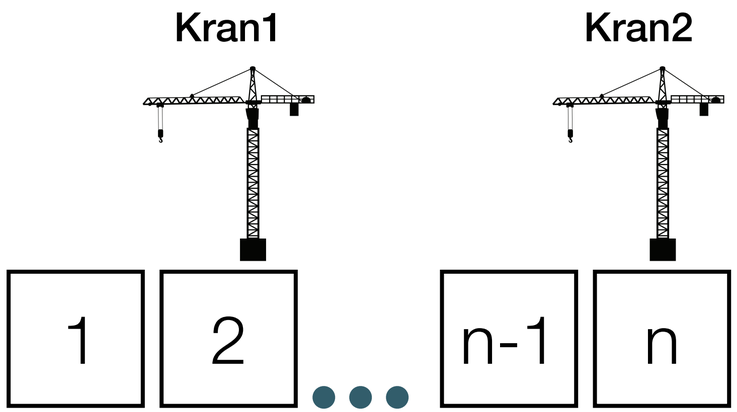
Min {$\mathbf{f_a(a,b,n)}$, $\mathbf{f_b(a,b,1)}$} sind die Optimalen Kosten

„Wenn Beide Kräne ihre Aufgaben abgearbeitet haben, müssen sie noch auf ihre Endposition fahren. Dann steht in $\mathbf{f_a(a,b,n)}$ oder $\mathbf{f_b(a,b,1)}$ die optimale Zeit“

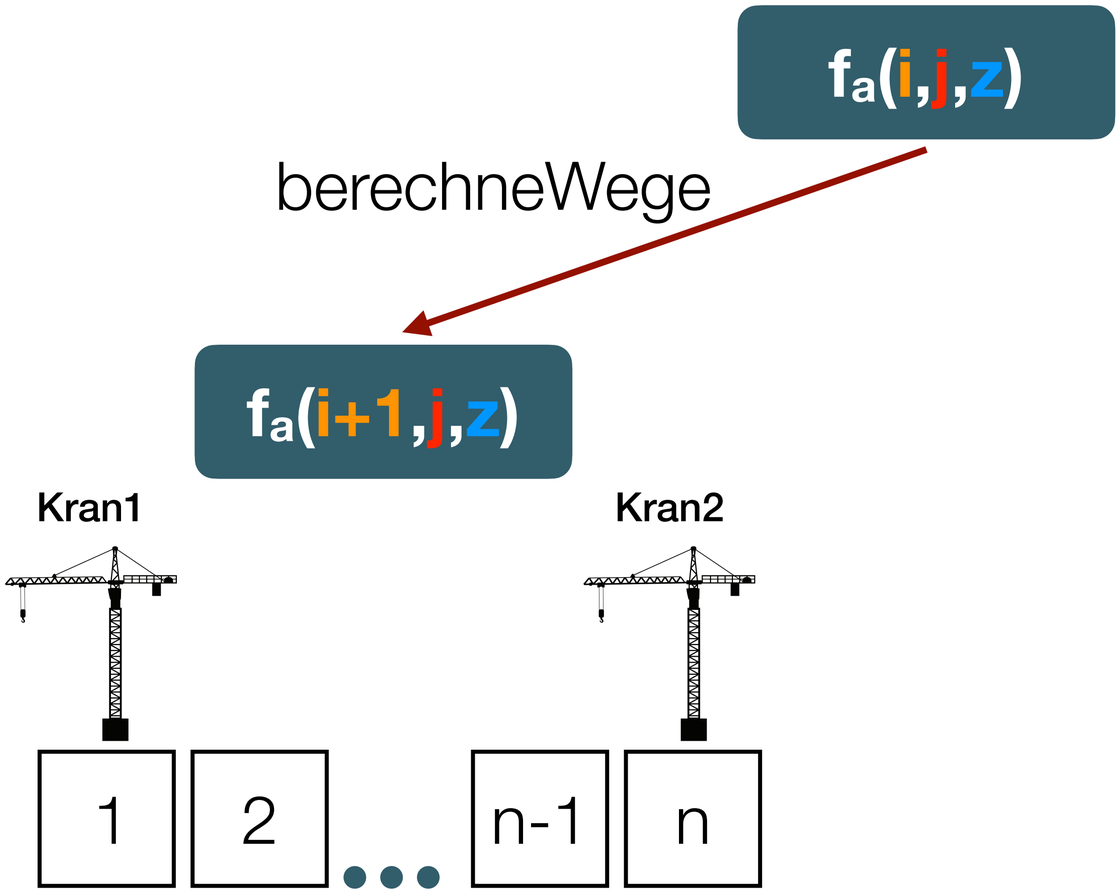


Mathematische Modellierung : Modell als Graph

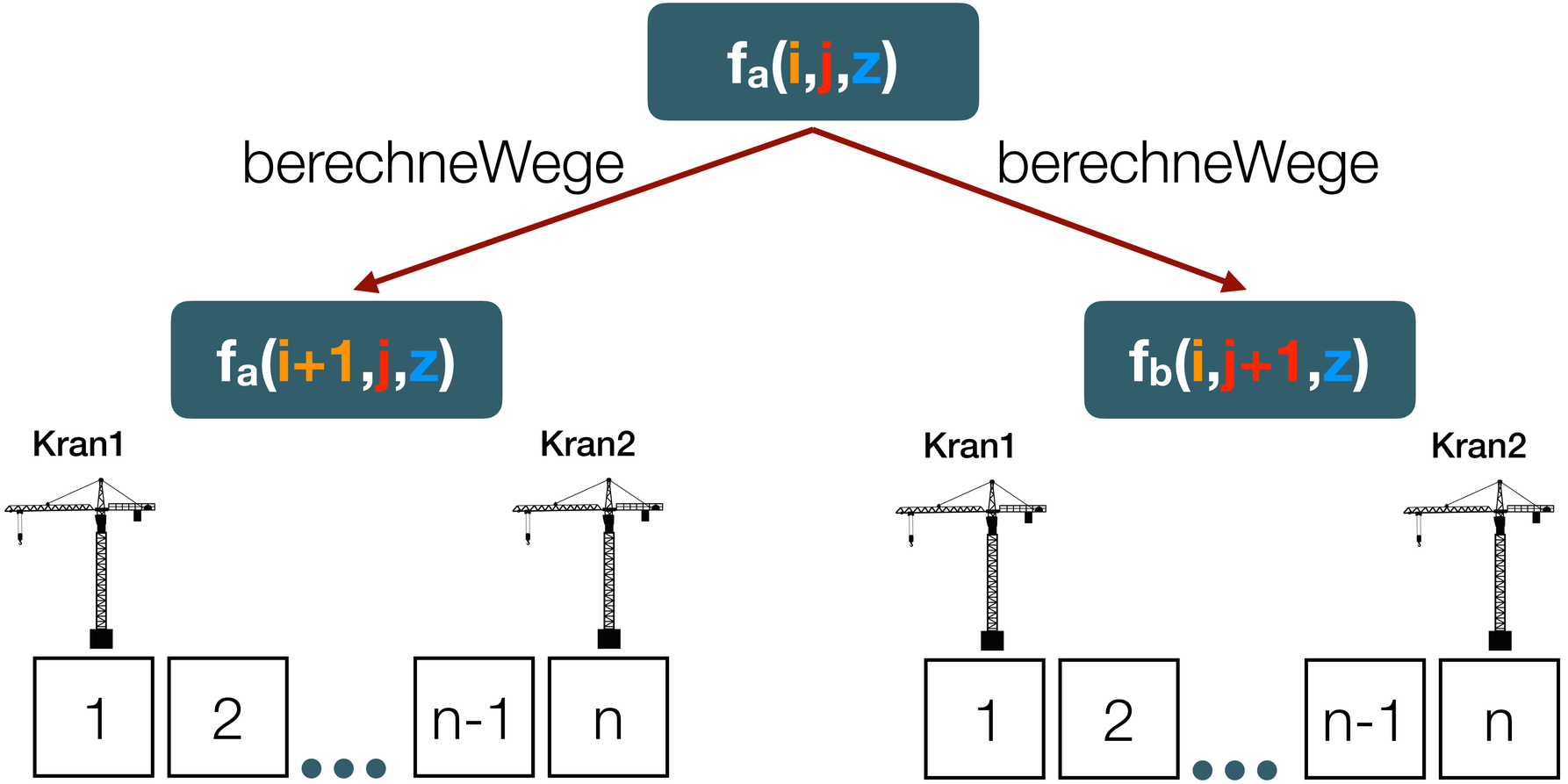
$f_a(i,j,z)$



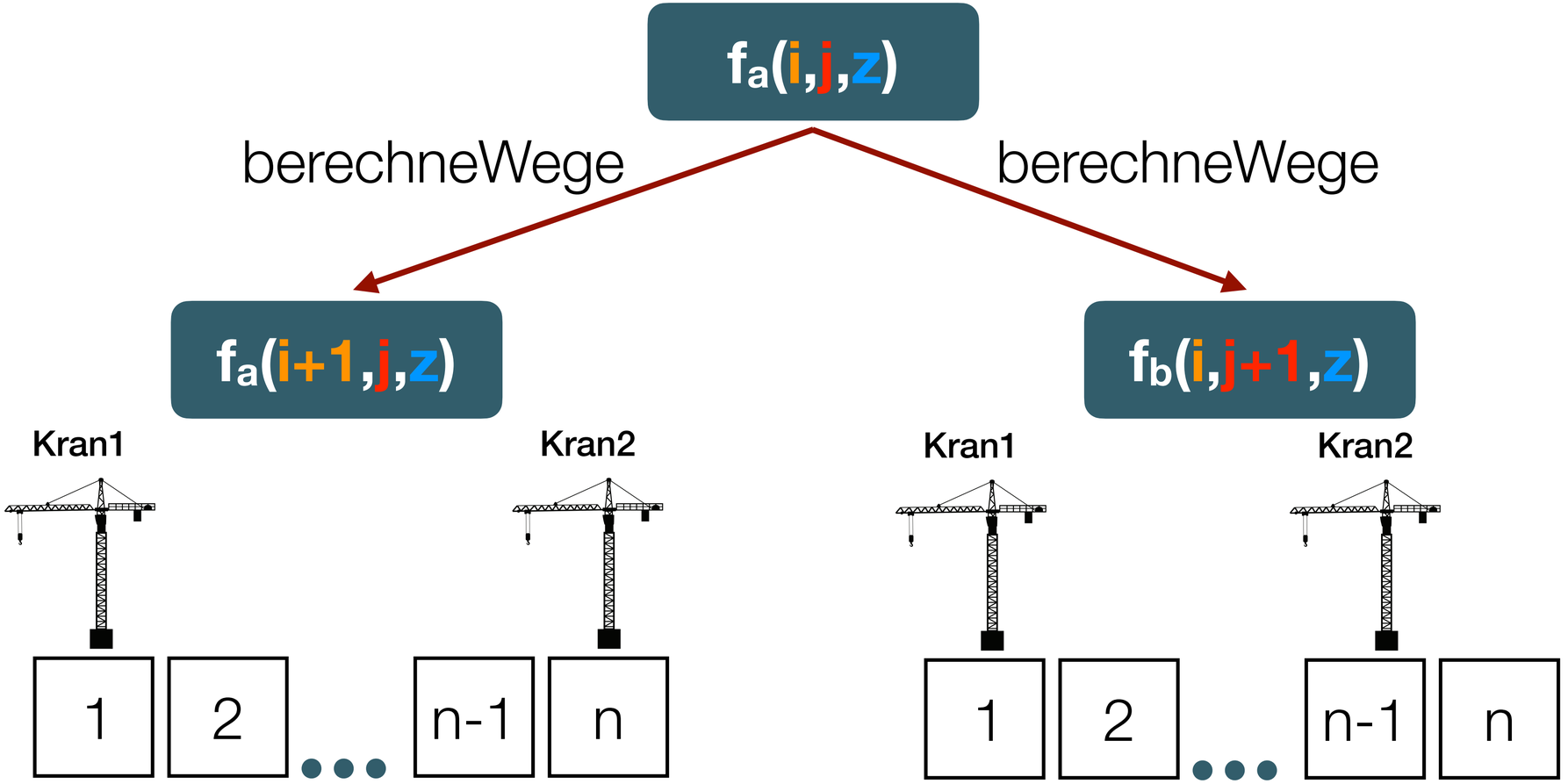
Mathematische Modellierung : Modell als Graph



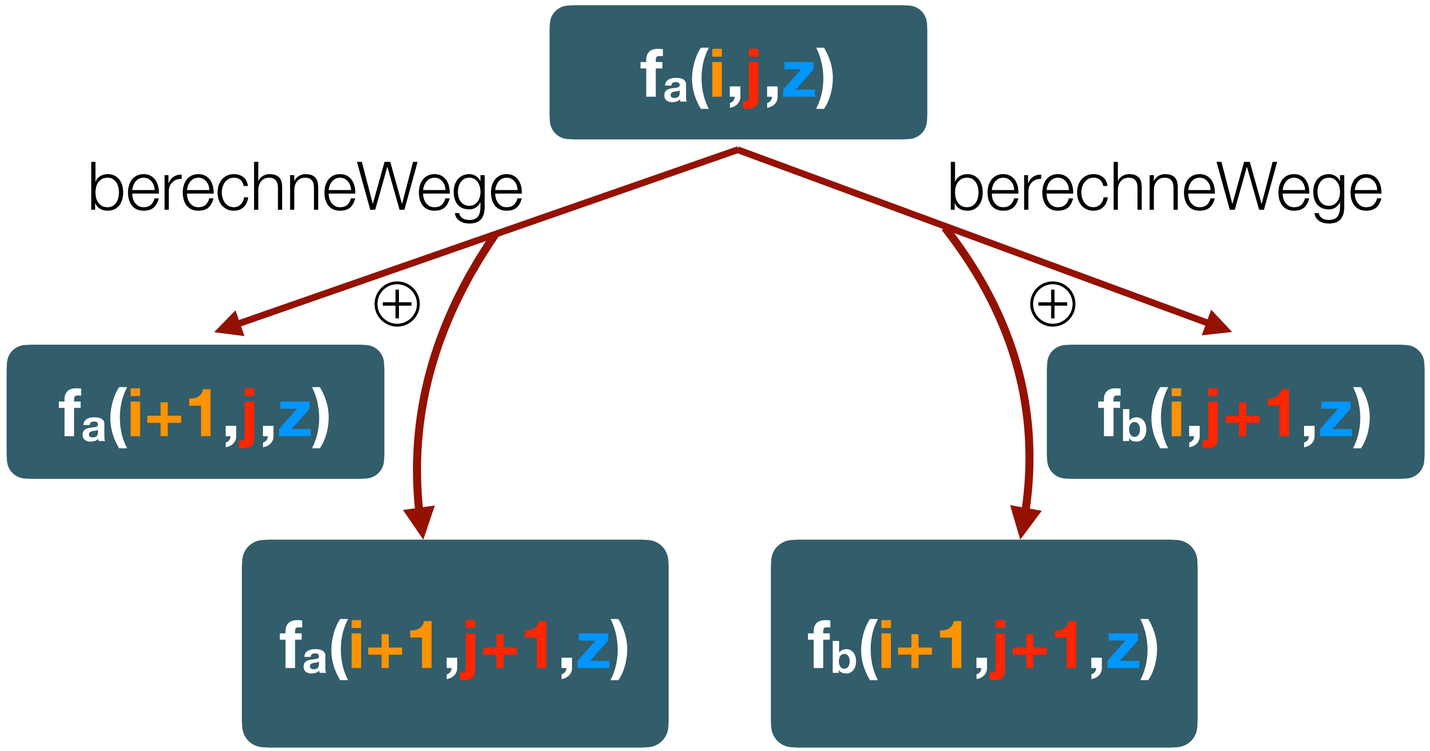
Mathematische Modellierung : Modell als Graph



Mathematische Modellierung : Modell als Graph

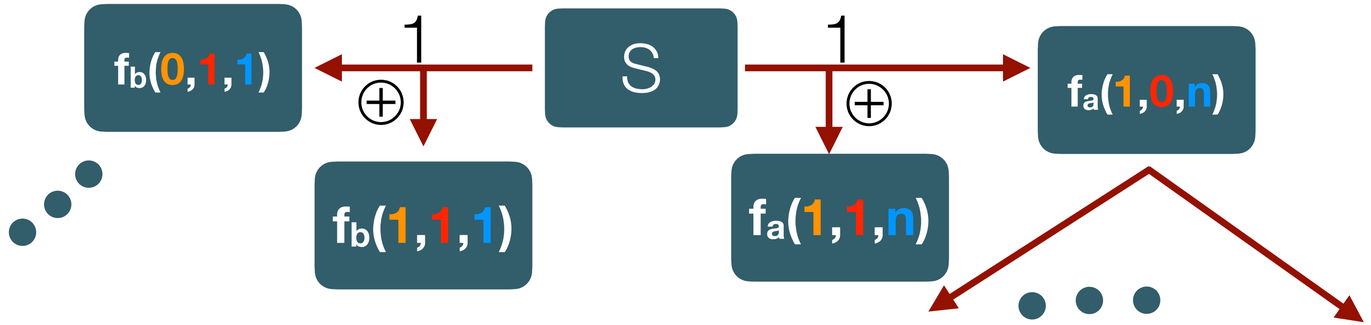


Mathematische Modellierung : Modell als Graph



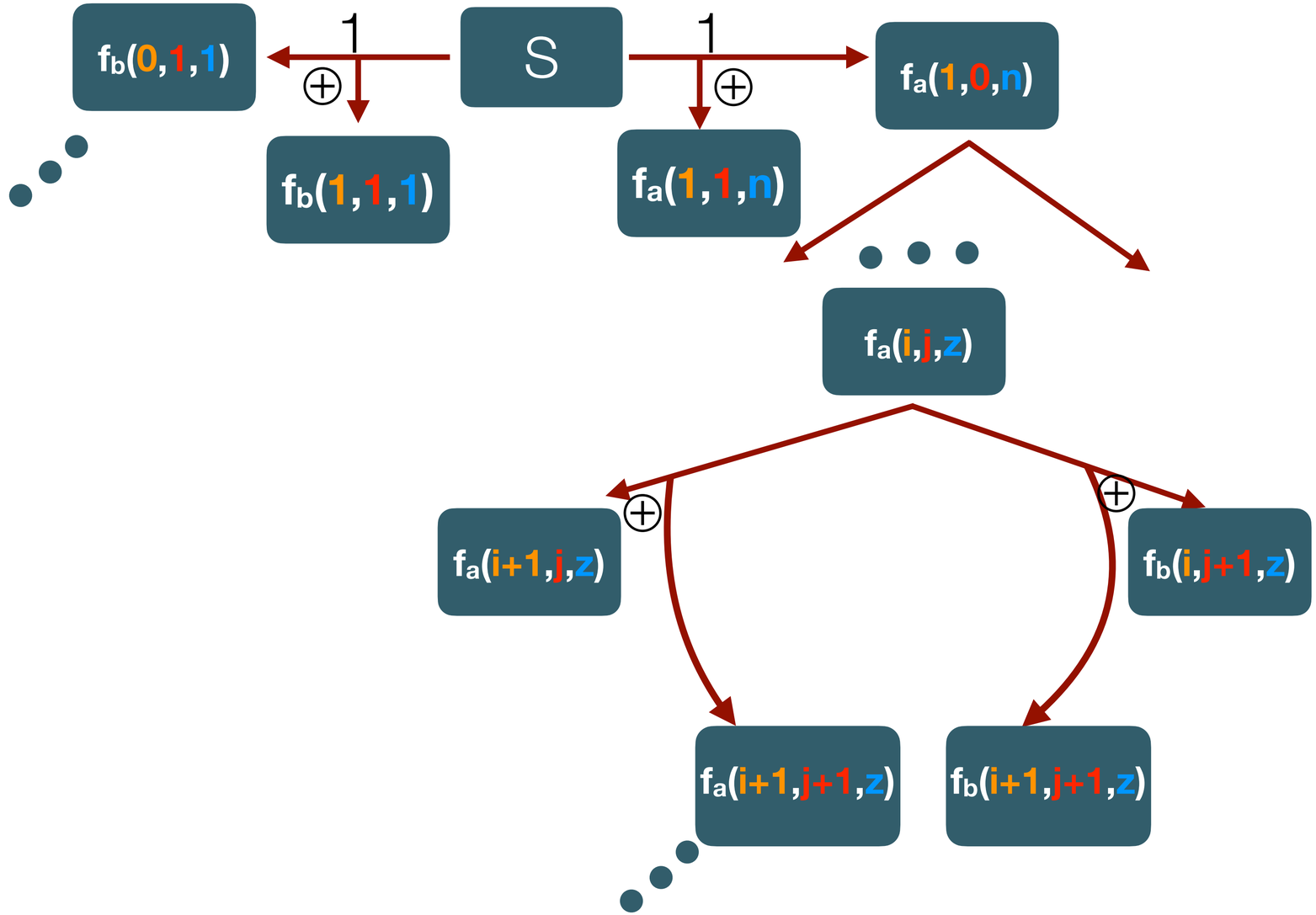
⊕ = Exklusives Oder

Mathematische Modellierung : Modell als Graph



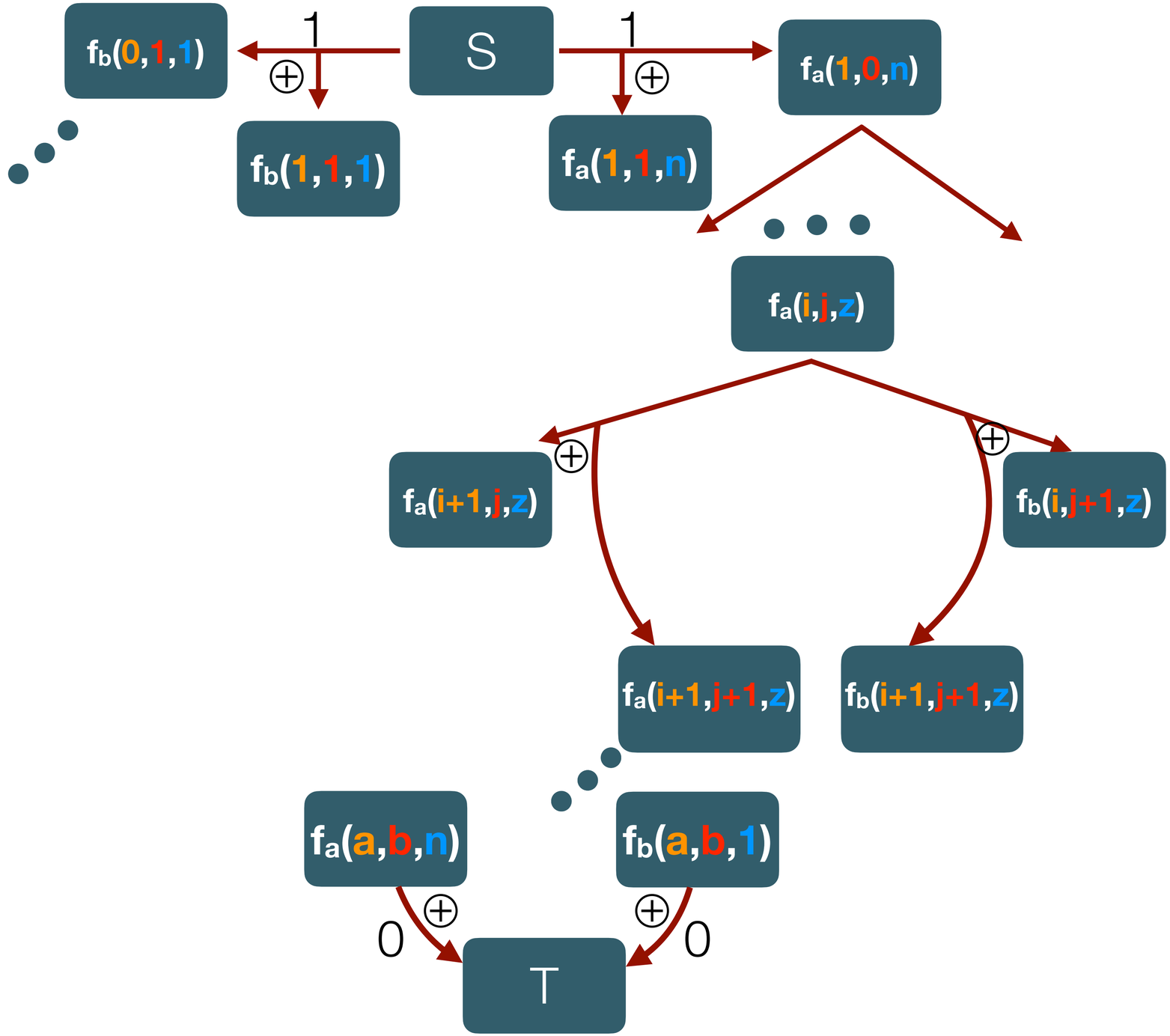
⊕ = Exklusives Oder

Mathematische Modellierung : Modell als Graph



⊕ = Exklusives Oder

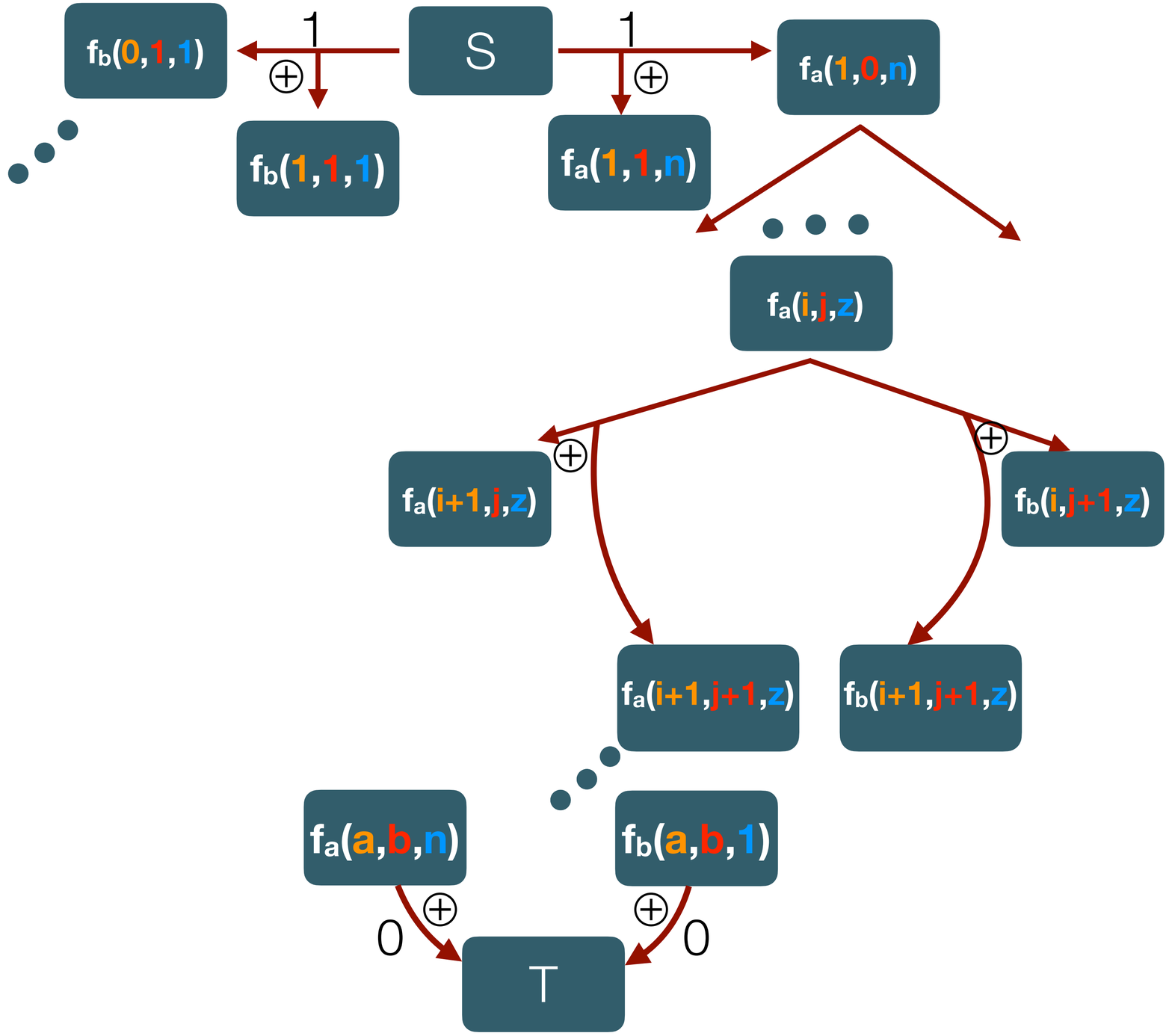
Mathematische Modellierung : Modell als Graph



⊕ = Exklusives Oder

Mathematische Modellierung : Modell als Graph

Jeder Knoten enthält die minimalen Kosten, um auf diese Konfiguration zu kommen



⊕ = Exklusives Oder

Wichtig zu beachten:

- Die Berechne-Wege-Funktion berechnet, dass Kran 1 eine Aufgabe abarbeitet und Kran 2 innerhalb dieser Zeit zu seiner nächsten Aufgabe läuft und ggf. diese abarbeitet. (Analog andersherum)

Wichtig: Die Beschränkungen müssen immer eingehalten werden!

- Kran 2 darf Kran 1 nicht Kreuzen, oder auf dem selben Feld stehen.

- Es muss eine minimale Zeit errechnet werden für die folgende Konfiguration.

Laufzeit

- Anzahl an Knoten **K**: $|a| \times |b| \times |n| \times 2 \leq 50 \times 50 \times 2000 \times 2 = 10.000.000$ Knoten

Laufzeit

- Anzahl an Knoten **K**: $|a| \times |b| \times |n| \times 2 \leq 50 \times 50 \times 2000 \times 2 = 10.000.000$ Knoten

- Anzahl Berechnungen: $|\mathbf{K}| \times 2$ = Anzahl der berechneten Kanten

Laufzeit

- Anzahl an Knoten **K**: $|a| \times |b| \times |n| \times 2 \leq 50 \times 50 \times 2000 \times 2 = 10.000.000$ Knoten

- Anzahl Berechnungen: $|\mathbf{K}| \times 2$ = Anzahl der berechneten Kanten

  - Zuerst alle Knoten Initialisieren und die Kosten auf $+\infty$ setzen

Laufzeit

- Anzahl an Knoten **K**: $|a| \times |b| \times |n| \times 2 \leq 50 \times 50 \times 2000 \times 2 = 10.000.000$ Knoten
- Anzahl Berechnungen: $|\mathbf{K}| \times 2$ = Anzahl der berechneten Kanten
  - Zuerst alle Knoten Initialisieren und die Kosten auf $+\infty$ setzen
  - Für jeden Knoten $\mathbf{f_{a/b}(i,j,x)}$ berechne die zwei ausgehenden Kanten zu:
    - $\mathbf{f_a(i+1, j, x) \oplus f_a(i+1, j+1, x)}$
    - $\mathbf{f_b(i, j+1, x) \oplus f_b(i+1, j+1, x)}$

Laufzeit

$$\text{Laufzeit}_{\text{Algorithmus}} \in O(|\mathbf{K}|) \text{ , wobei } |\mathbf{K}| \text{ die Anzahl der Knoten ist}$$

Pseudocode

Initialize: $f_a(1,1,n) = 1 = f_b(1,1,1)$

FOR i=1 **TO** a:

    FOR j=1 **TO** b:

        FOR x = n **DOWN TO** p_{i+1} :

            berechne_wege(…)

        FOR x = 1 **TO** q_j:

            berechne_wege(…)

Return min{$f_a(a,b,n)$, $f_b(a,b,1)$}

Pseudocode: berechnet Wege

berechne_wege(…)

while(Crane_1 **!=** finished **)** :

    ΔtimeSteps = | p_i - p_{i+1} |

    positionGantryCrane_2 = positionGantryCrane_2 ± ΔtimeSteps

    if positionGantryCrane_2 == q_{j+1}

        j = j + 1

    else

        positionGantryCrane_2 = positionGantryCrane_2 ± (ΔtimeSteps + 1)

    $timeSteps_{old}$ = f_a(i+1,j, Position_kran_2)

    f_a(i+1,j, positionGantryCrane_2) = Min{$timeSteps_{old}$, Timesteps + ΔtimeSteps + 1}

while(Crane_2 **!=** finished **)**: …

Fallstricke und Hinweise

- o.B.d.A: Kran 1 macht Aufgabe, Kran 2 läuft
  - Kran 2 kann verdrängt werden, falls er im Weg steht
  - Kran 2 kann begrenzt viele Schritte in die Richtung seiner nächsten Aufgabe fahren
  - Kran 2 kann innerhalb der Zeit, die Kran 1 benötigt, eine Aufgabe erledigen

Fallstricke und Hinweise

- o.B.d.A: Kran 1 macht Aufgabe, Kran 2 läuft
  - Kran 2 kann verdrängt werden, falls er im Weg steht
  - Kran 2 kann begrenzt viele Schritte in die Richtung seiner nächsten Aufgabe fahren
  - Kran 2 kann innerhalb der Zeit, die Kran 1 benötigt, eine Aufgabe erledigen
- Knoten in einem 3-Dimensionalen Array speichern Table[**i**][**j**][**x**] mit je zwei Werten

Fallstricke und Hinweise

- o.B.d.A: Kran 1 macht Aufgabe, Kran 2 läuft
  - Kran 2 kann verdrängt werden, falls er im Weg steht
  - Kran 2 kann begrenzt viele Schritte in die Richtung seiner nächsten Aufgabe fahren
  - Kran 2 kann innerhalb der Zeit, die Kran 1 benötigt, eine Aufgabe erledigen
- Knoten in einem 3-Dimensionalen Array speichern Table[**i**][**j**][**x**] mit je zwei Werten
- Knoten müssen in der Reihenfolge der Aufgaben berechnet werden