Seminar02-1-Annotated

State Space Search

Dr. Collin F. Lynch

AI Academy: North Carolina State University

AI Academy

NC STATE UNIVERSITY

Seminar02-1-Annotated

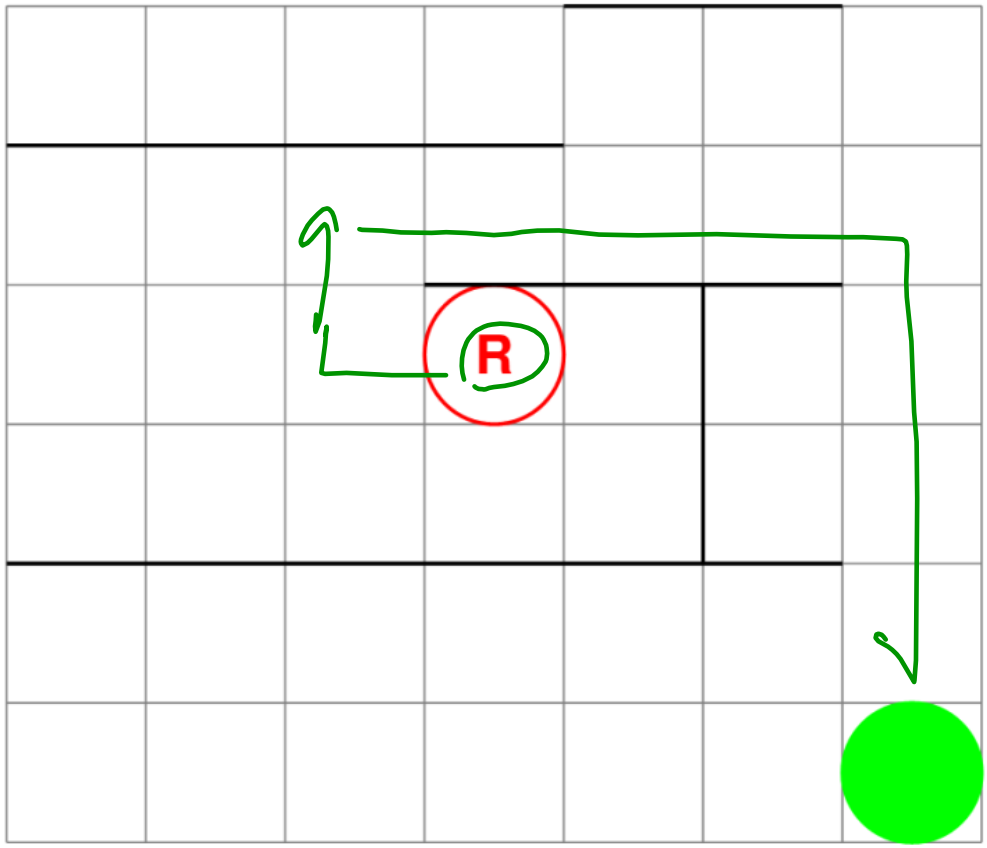
Agenda

- Search Problems
- Uninformed Search
- Real Costs and Goals
- A*
- RBFS

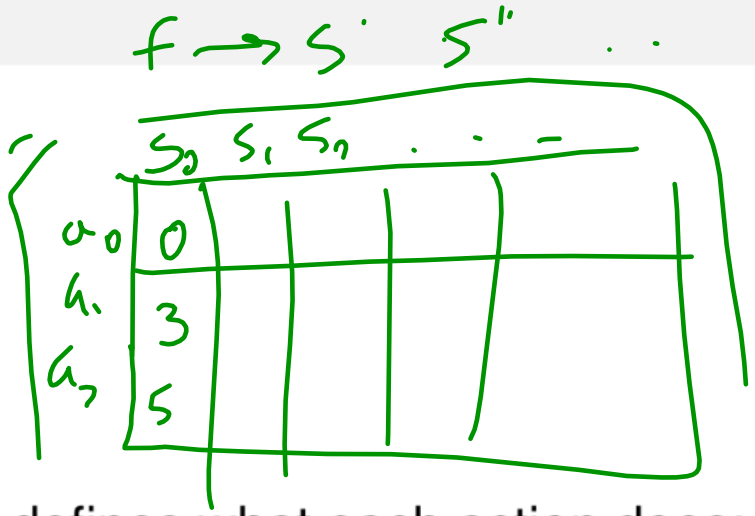
Search Problems

Seminar02-1-Annotated

VacBot



Problem Definition



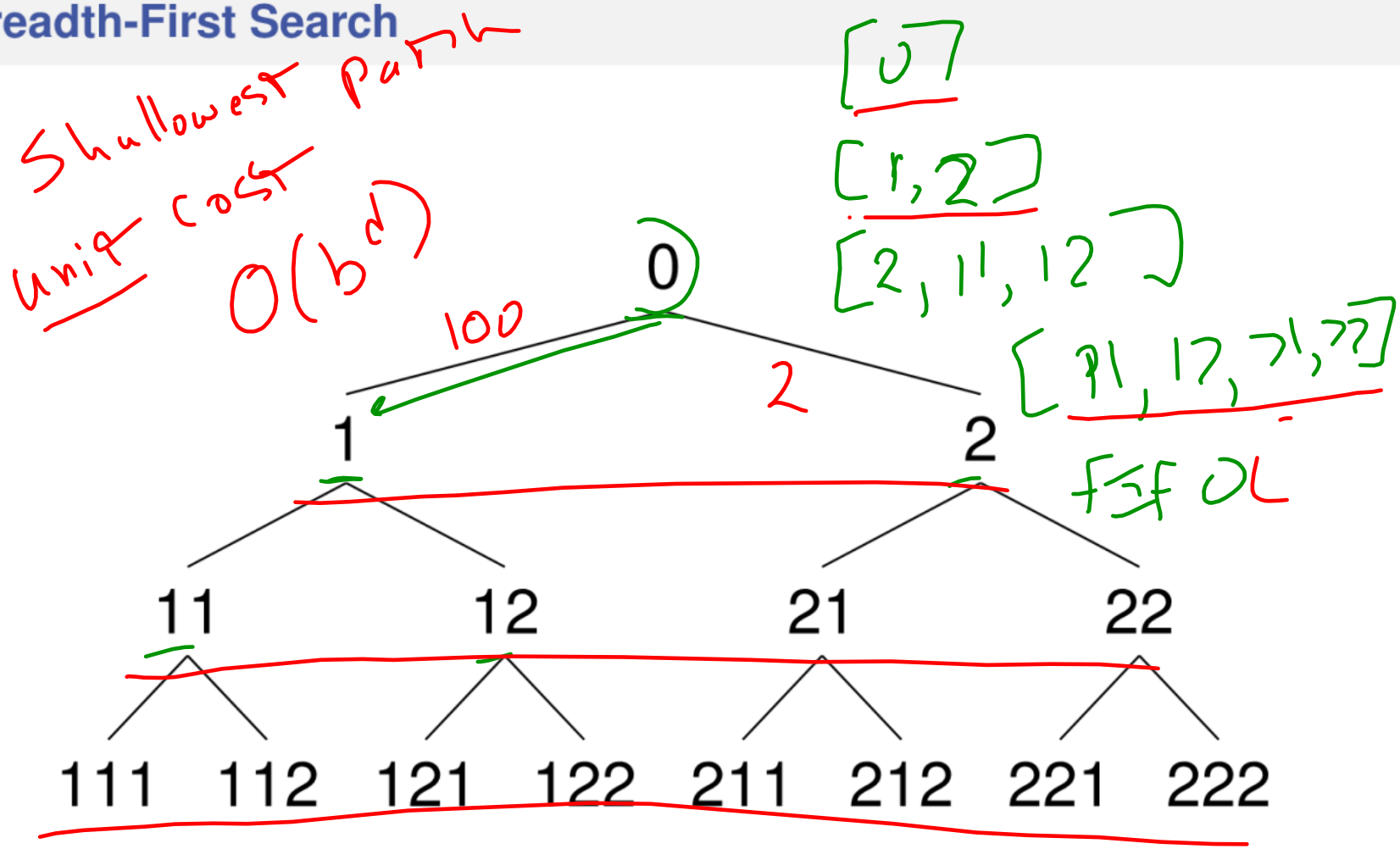
1. The *Start* or *Initial* State.
2. The available *actions*.
3. The *Transition Model* which defines what each action does:

$$f : S_i \oplus A_j \rightarrow S_k \tag{1}$$

4. The *goal test* (sometimes *goal state(s)*).
5. The *path cost* or *cost function*.

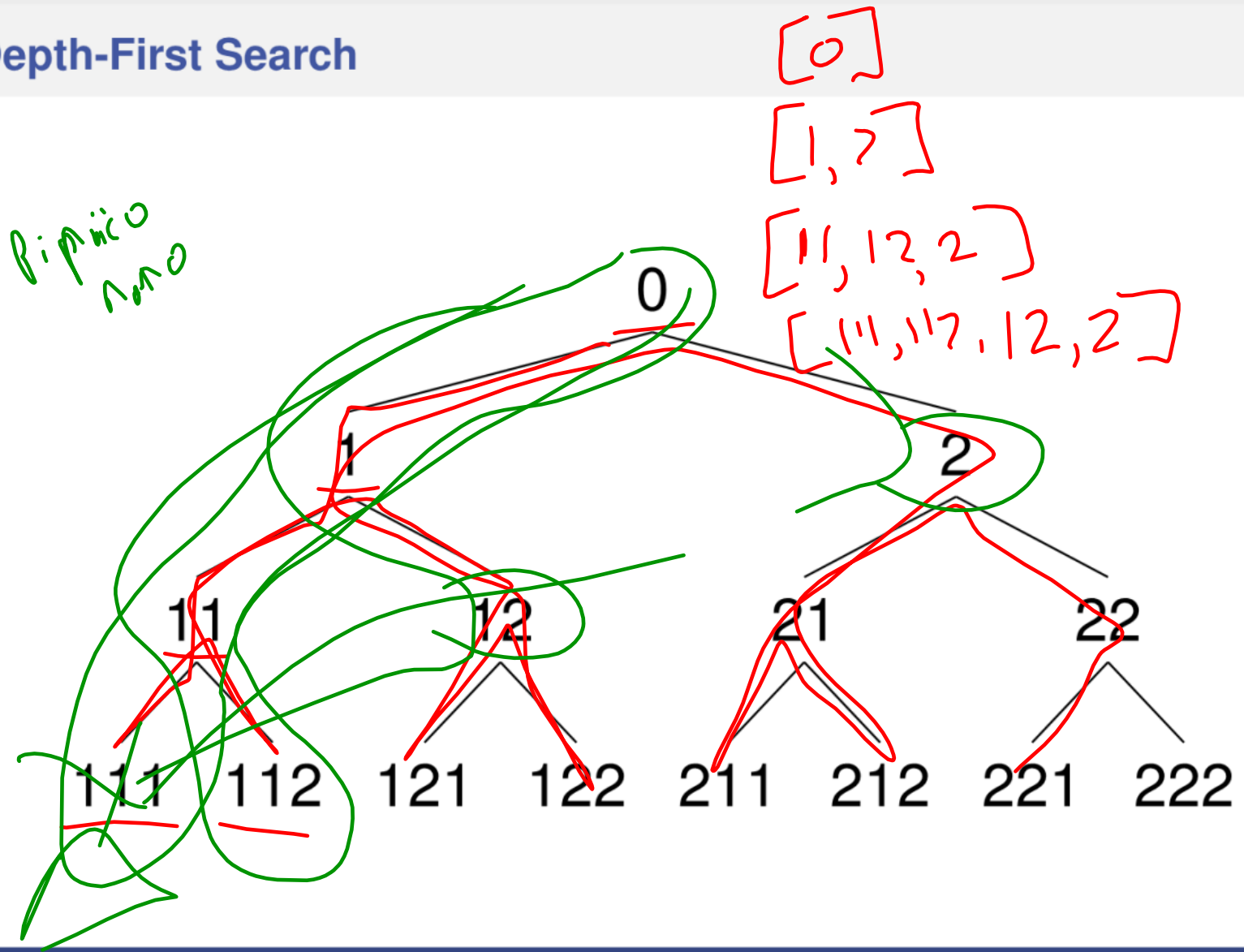
Uninformed Search

Breadth-First Search



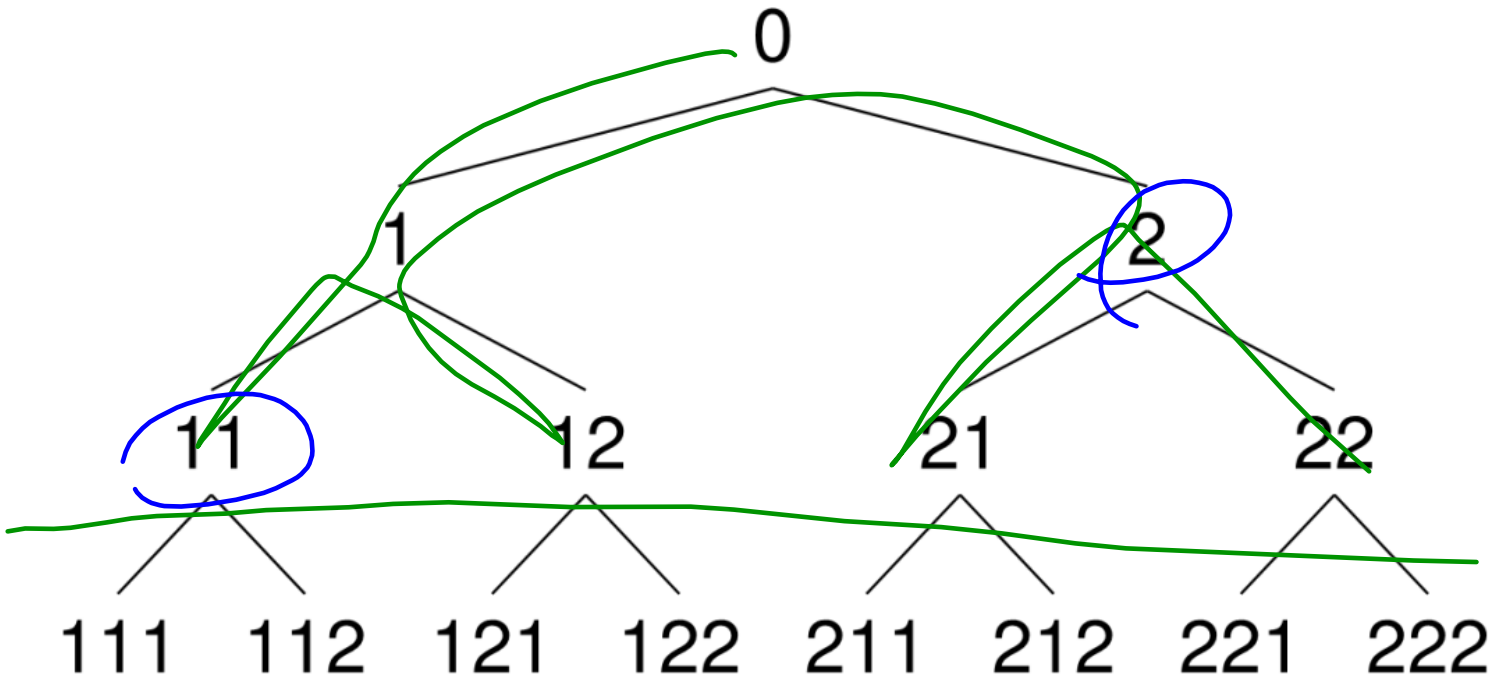
Seminar02-1-Annotated

Depth-First Search



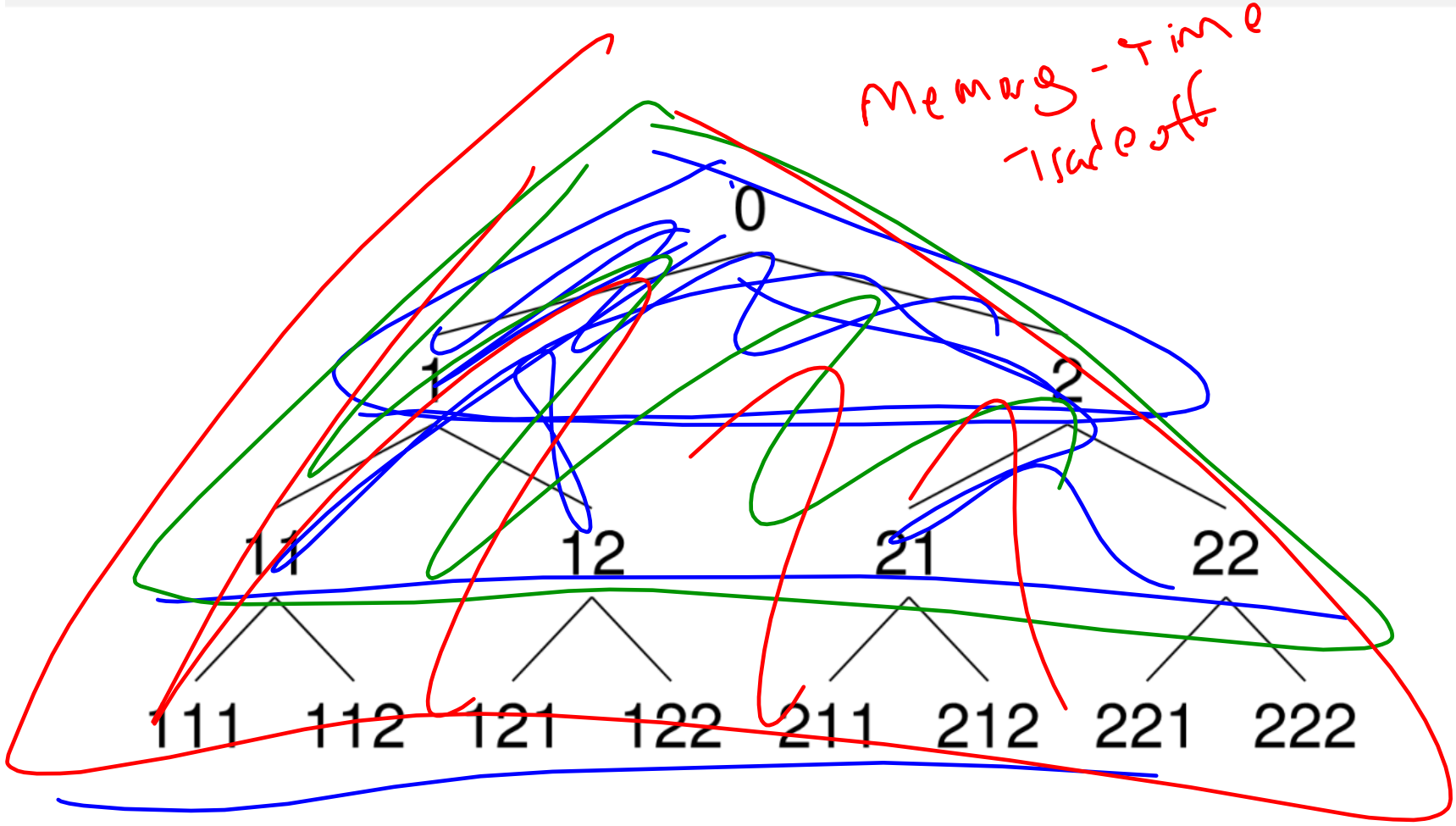
Depth-Limited Search (Fixed Diameter)

Relative Error



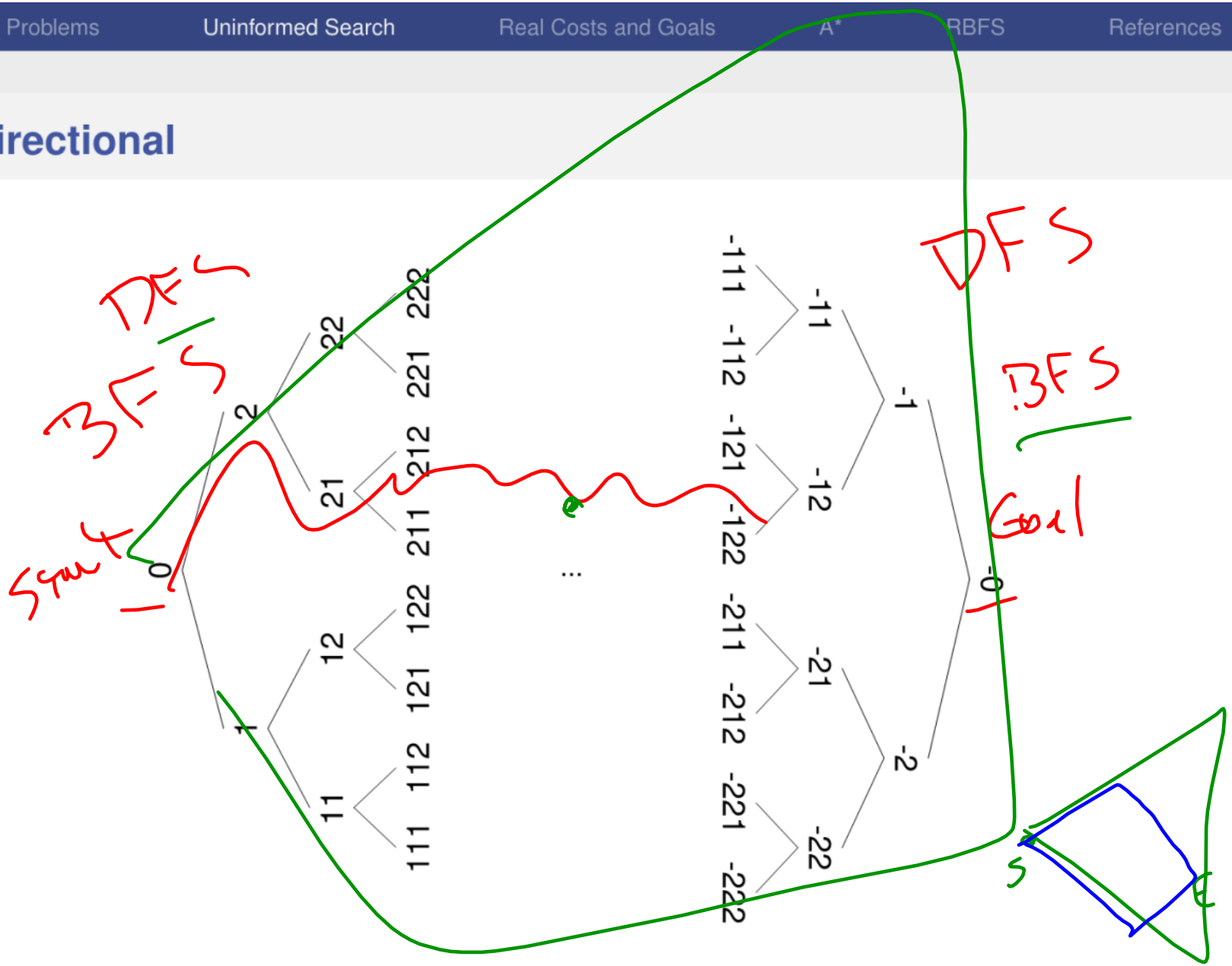
Seminar02-1-Annotated

Iterative-Deepening (Soft Diameter)



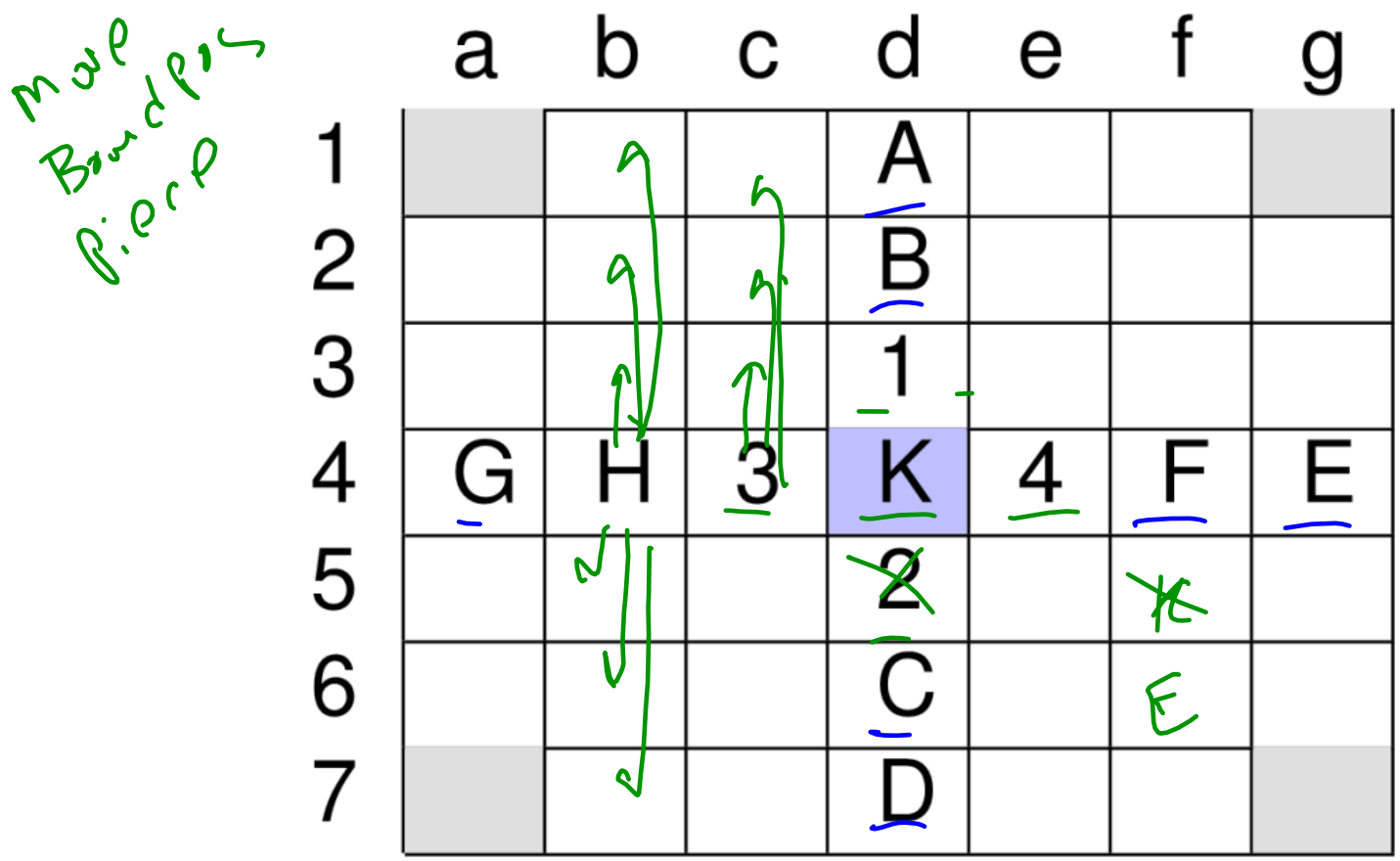
Seminar02-1-Annotated

Bidirectional

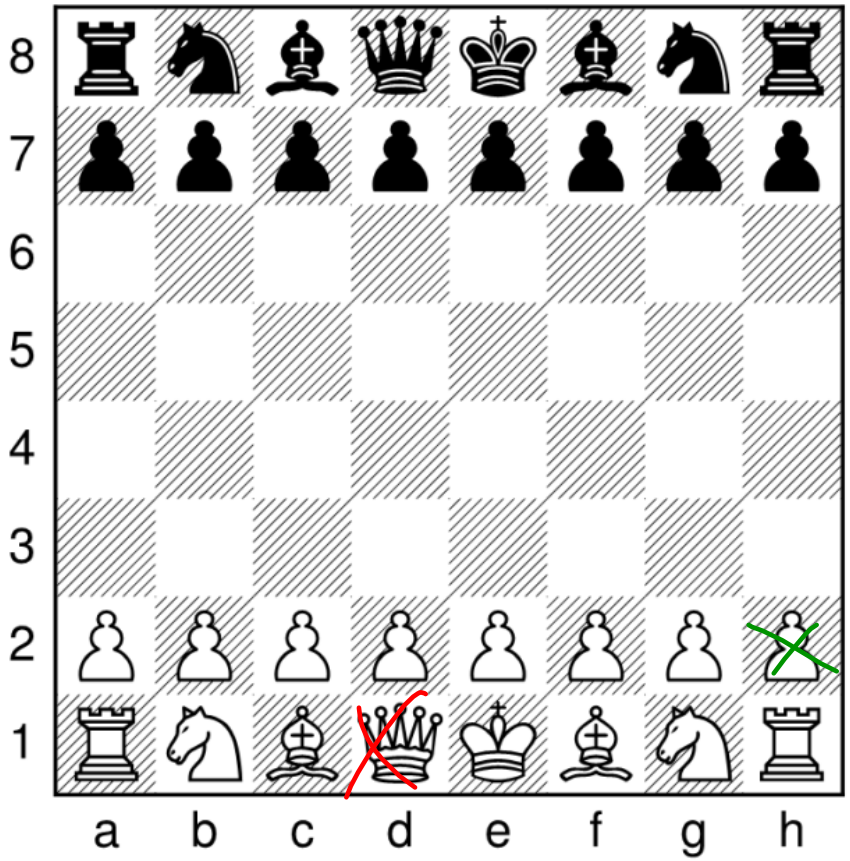


Real Costs and Goals

Clear Costs?



Differential Costs



Seminar02-1-Annotated

Calendar

2013 Ferienkalender

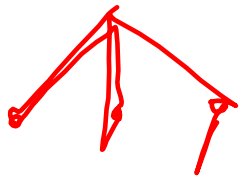
Januar	Februar	März	April	Mai	Juni
1 Di Neujahr	1 Fr	1 Fr	1 Mo Ostermontag	1 Mi Tag der Arbeit	1 Sa
2 Mi	2 Sa	2 Sa	2 Di	2 Do	2 So
3 Do	3 So	3 So	3 Mi	3 Fr	3 Mo
4 Fr	4 Mo	4 Mo	4 Do	4 Sa	4 Di
5 Sa	5 Di	5 Di	5 Fr	5 So	5 Mi
6 So Heilige Drei Könige	6 Mi	6 Mi	6 Sa	6 Mo	6 Do
7 Mo	7 Do	7 Do	7 So	7 Di	7 Fr
8 Di	8 Fr	8 Fr	8 Mo	8 Mi	8 Sa
9 Mi	9 Sa	9 Sa	9 Di	9 Do Christi Himmelfahrt	9 So
10 Do	10 So	10 So	10 Mi	10 Fr	10 Mo
11 Fr	11 Mo	11 Mo	11 Do	11 Sa	11 Di
12 Sa	12 Di	12 Di	12 Fr	12 So	12 Mi
13 So	13 Mi	13 Mi	13 Sa	13 Mo	13 Do
14 Mo	14 Do	14 Do	14 So	14 Di	14 Fr
15 Di	15 Fr	15 Fr	15 Mo	15 Mi	15 Sa
16 Mi	16 Sa	16 Sa	16 Di	16 Do	16 So
17 Do	17 So	17 So	17 Mi	17 Fr	17 Mo
18 Fr	18 Mo	18 Mo	18 Do	18 Sa	18 Di
19 Sa	19 Di	19 Di	19 Fr	19 So Pfingstsonntag	19 Mi
20 So	20 Mi	20 Mi	20 Sa	20 Mo Pfingstmontag	20 Do
21 Mo	21 Do	21 Do	21 So	21 Di	21 Fr
22 Di	22 Fr	22 Fr	22 Mo	22 Mi	22 Sa
23 Mi	23 Sa	23 Sa	23 Di	23 Do	23 So
24 Do	24 So	24 So	24 Mi	24 Fr	24 Mo
25 Fr	25 Mo	25 Mo	25 Do	25 Sa	25 Di
26 Sa	26 Di	26 Di	26 Fr	26 So	26 Mi
27 So	27 Mi	27 Mi	27 Sa	27 Mo	27 Do
28 Mo	28 Do	28 Do	28 So	28 Di	28 Fr
29 Di		29 Fr Karfreitag	29 Mo	29 Mi	29 Sa
30 Mi		30 Sa	30 Di	30 Do Fronleichnam	30 So
31 Do		31 So Ostersonntag		31 Fr	

http://www.texample.net/

Calendar Parts

- **Actions:**
  - Add a booking.
  - Move a booking?
  - Shrink a booking?
- **Transition Function**:
  - Reduce the number of missing bookings.
  - Add individual constraints?

Calendar Parts



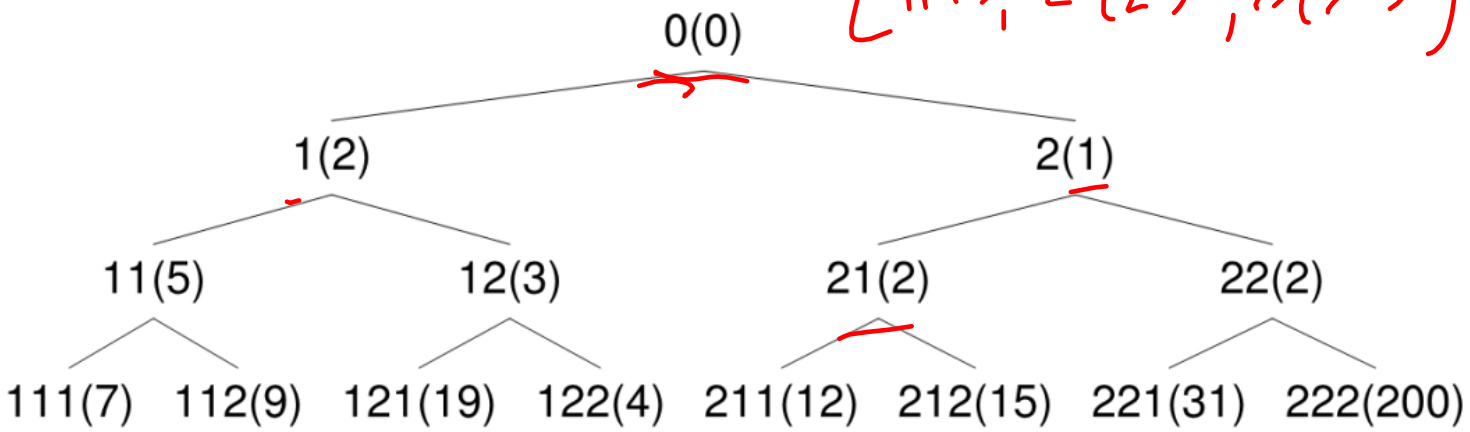
- **Goal Test:**
  - Everyone Booked?
  - No other booking possible?

Retrospective

- **Cost function:**
  - Number of schedules made?
  - Number of conflicts?
  - Number of people forced to change?

offline

Cost-Sensitive Search



Seminar02-1-Annotated

~~Informed~~ ~~Heuristic~~ Future Search

Seminar02-1-Annotated

Problems to Consider: Packet Routing

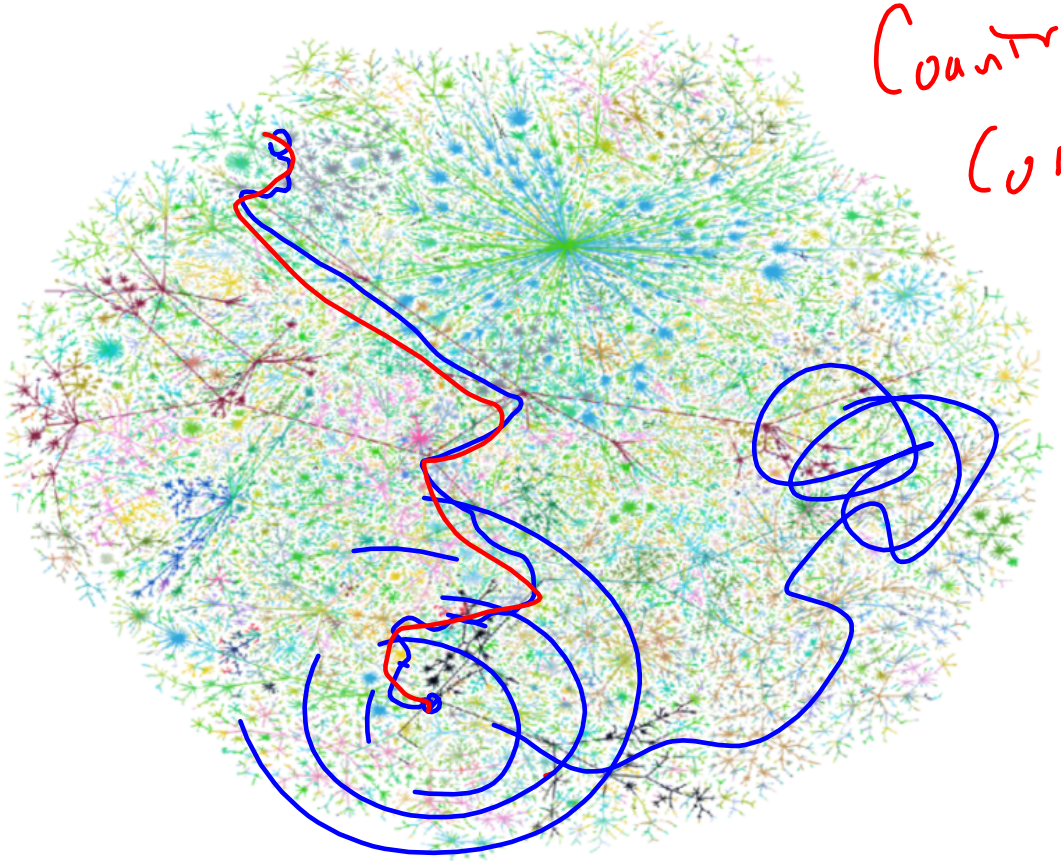


Image credit:

http://aminotes.tumblr.com/post/13219118418/the-maps-of-the-internet

Problems to Consider: Calendar

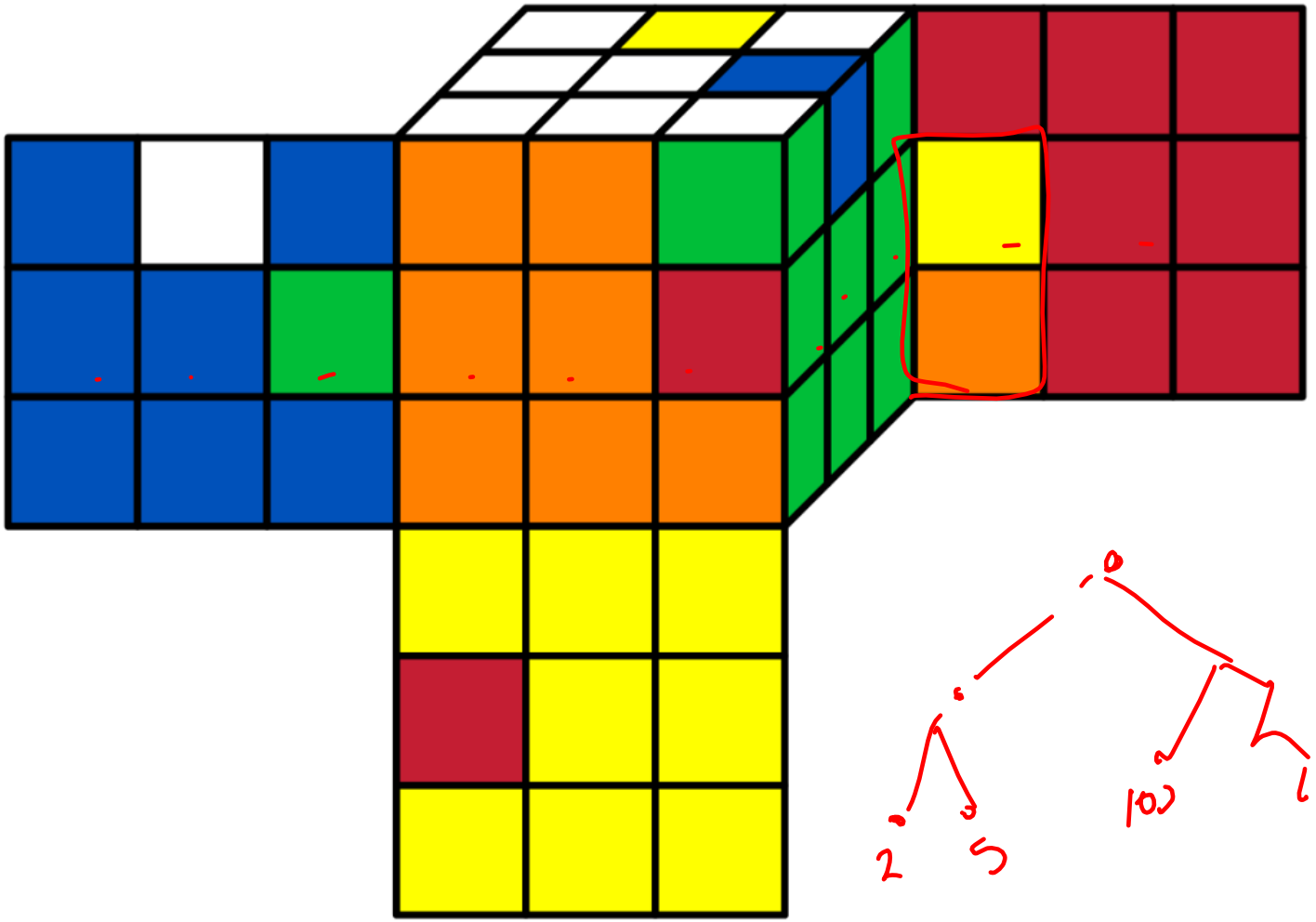
2013 Ferienkalender

Januar	Februar	März	April	Mai	Juni
1 Di Neujahr	1 Fr	1 Fr	1 Mo Ostermontag	1 Mi Tag der Arbeit	1 Sa
2 Mi	2 Sa	2 Sa	2 Di	2 Do	2 So
3 Do	3 So	3 So	3 Mi	3 Fr	3 Mo
4 Fr	4 Mo	4 Mo	4 Do	4 Sa	4 Di
5 Sa	5 Di	5 Di	5 Fr	5 So	5 Mi
6 So Heilige Drei Könige	6 Mi	6 Mi	6 Sa	6 Mo	6 Do
7 Mo	7 Do	7 Do	7 So	7 Di	7 Fr
8 Di	8 Fr	8 Fr	8 Mo	8 Mi	8 Sa
9 Mi	9 Sa	9 Sa	9 Di	9 Do Christi Himmelfahrt	9 So
10 Do	10 So	10 So	10 Mi	10 Fr	10 Mo
11 Fr	11 Mo	11 Mo	11 Do	11 Sa	11 Di
12 Sa	12 Di	12 Di	12 Fr	12 So	12 Mi
13 So	13 Mi	13 Mi	13 Sa	13 Mo	13 Do
14 Mo	14 Do	14 Do	14 So	14 Di	14 Fr
15 Di	15 Fr	15 Fr	15 Mo	15 Mi	15 Sa
16 Mi	16 Sa	16 Sa	16 Di	16 Do	16 So
17 Do	17 So	17 So	17 Mi	17 Fr	17 Mo
18 Fr	18 Mo	18 Mo	18 Do	18 Sa	18 Di
19 Sa	19 Di	19 Di	19 Fr	19 So Pfingstsonntag	19 Mi
20 So	20 Mi	20 Mi	20 Sa	20 Mo Pfingstmontag	20 Do
21 Mo	21 Do	21 Do	21 So	21 Di	21 Fr
22 Di	22 Fr	22 Fr	22 Mo	22 Mi	22 Sa
23 Mi	23 Sa	23 Sa	23 Di	23 Do	23 So
24 Do	24 So	24 So	24 Mi	24 Fr	24 Mo
25 Fr	25 Mo	25 Mo	25 Do	25 Sa	25 Di
26 Sa	26 Di	26 Di	26 Fr	26 So	26 Mi
27 So	27 Mi	27 Mi	27 Sa	27 Mo	27 Do
28 Mo	28 Do	28 Do	28 So	28 Di	28 Fr
29 Di		29 Fr Karfreitag	29 Mo	29 Mi	29 Sa
30 Mi		30 Sa	30 Di	30 Do Fronleichnam	30 So
31 Do		31 So Ostersonntag		31 Fr	

http://www.texample.net/

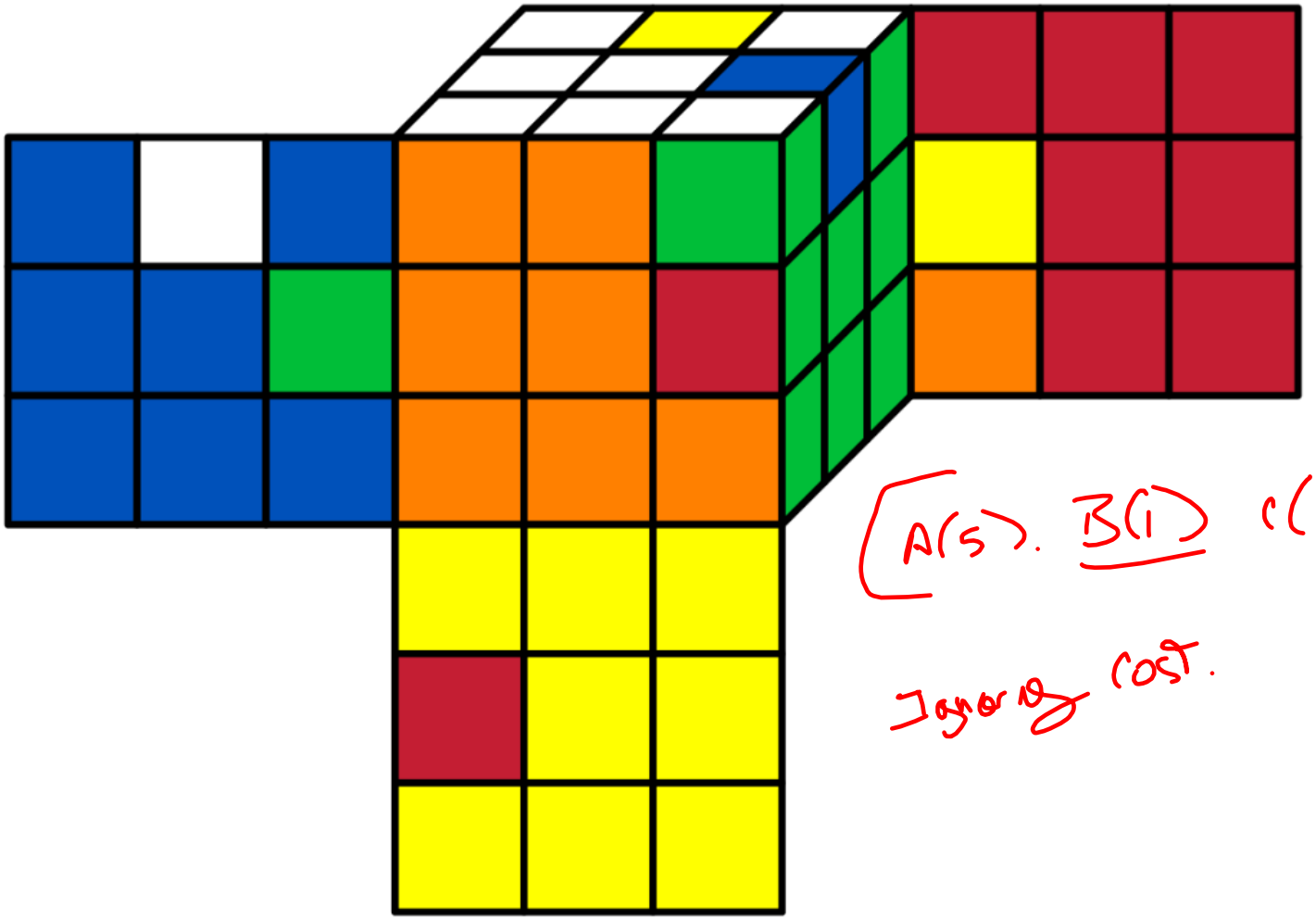
Seminar02-1-Annotated

Problems to Consider: Rubik's Cube



Seminar02-1-Annotated

Greedy Best First Search



[A(5), B(1), C(10)]

Ignoring cost.

Seminar02-1-Annotated

A*

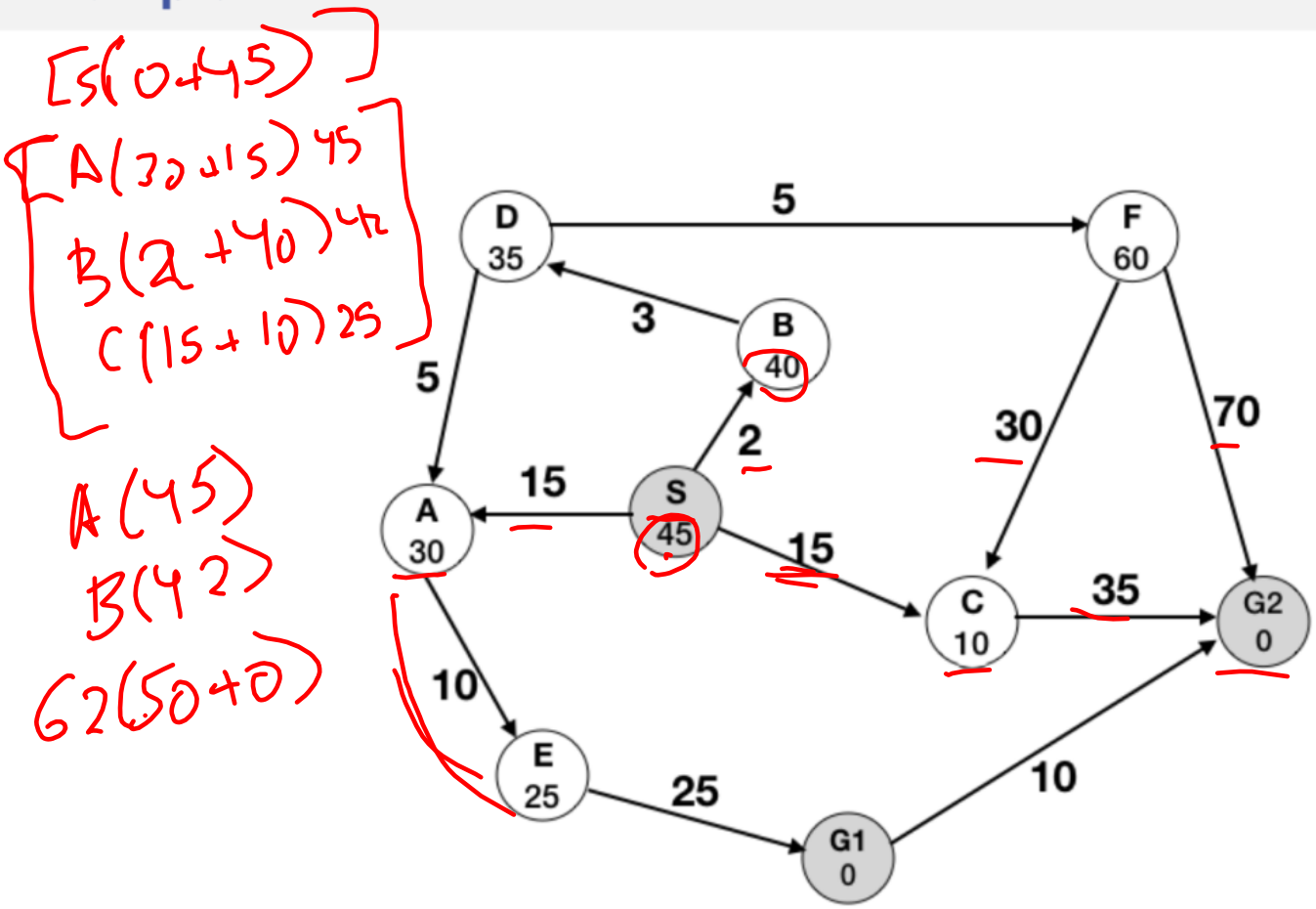
Complete & Optimal

UCS $f(n) = g(n)$
GBFS $f(n) = h(n)$

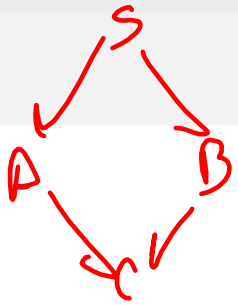
$$min_{f(n)} : f(n) = g(n) + h(n) \tag{2}$$

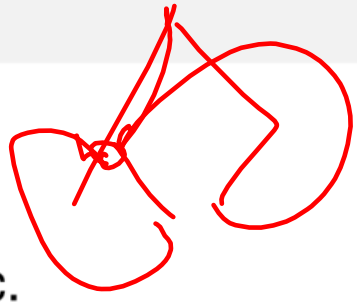
Seminar02-1-Annotated

Example



Optimality





- Tree Search: if $h(n)$ is *admissible* or optimistic.

$$\forall n : h(n) \leq (g(G_n) - g(n)) \tag{3}$$

- Graph Search: if $h(n)$ is *consistent* or monotonic.

$$\forall n : h(n) \leq c(n, a, n') + h(n') \tag{4}$$

Complete & Optimal

$g(n)$ $r(n,n') > 0$

- Complete? *Yes*
- Optimal? *Yes (possibly)*
- Cheap? **No** *(probably)*

Graph Search Optimality: Basic Sketch

- Node costs are nondecreasing.
- Thus we have the *Triangle Inequality*.
- Thus we only expand n when the optimal path to n is found.
- Thus nodes n_i expand in monotonic $g(n)$.
- Thus the first goal is the cheapest and no more expensive node will be expanded.

Efficiency

There exist two ways to assess A* efficiency.

- *Algorithmic Efficiency* i.e. O notation.
- *Relative Efficiency* compared to other search.

Algorithmic Efficiency

- If we let $h*(n)$ be the best *possible* heuristic for a problem (i.e. the actual cost to the goal).
- And we let $\Delta = h*(n) - h(n)$ be the absolute *error*.
- And we let $\epsilon = (h*(n) - h(n))/h(n)$ be the relative *error*.

Algorithmic Efficiency

- If we let $h*(n)$ be the best *possible* heuristic for a problem (i.e. the actual cost to the goal).
- And we let $\Delta = h*(n) - h(n)$ be the absolute *error*.
- And we let $\epsilon = (h*(n) - h(n))/h(n)$ be the relative *error*.
- Then in general A* is $O(b^{\epsilon d})$

Algorithmic Efficiency

- If we let $h*(n)$ be the best *possible* heuristic for a problem (i.e. the actual cost to the goal).
- And we let $\Delta = h*(n) - h(n)$ be the absolute *error*.
- And we let $\epsilon = (h*(n) - h(n))/h(n)$ be the relative *error*.
- Then in general A* is $O(b^{\epsilon d})$
- However:
  - If $|\Delta| = O(lgh*(n))$.
  - And the space is a tree.
    - THEN: it is polynomial.

Relative Efficiency

- A* is optimally efficient among algorithms of this type *for any given consistent heuristic.*
- It expands fewer nodes (give or take contour ties).
- **But** that number may be exponential in the length of the solution.
- And it thrashes when there are many near-optimal states.

Seminar02-1-Annotated

Efficiency (2)

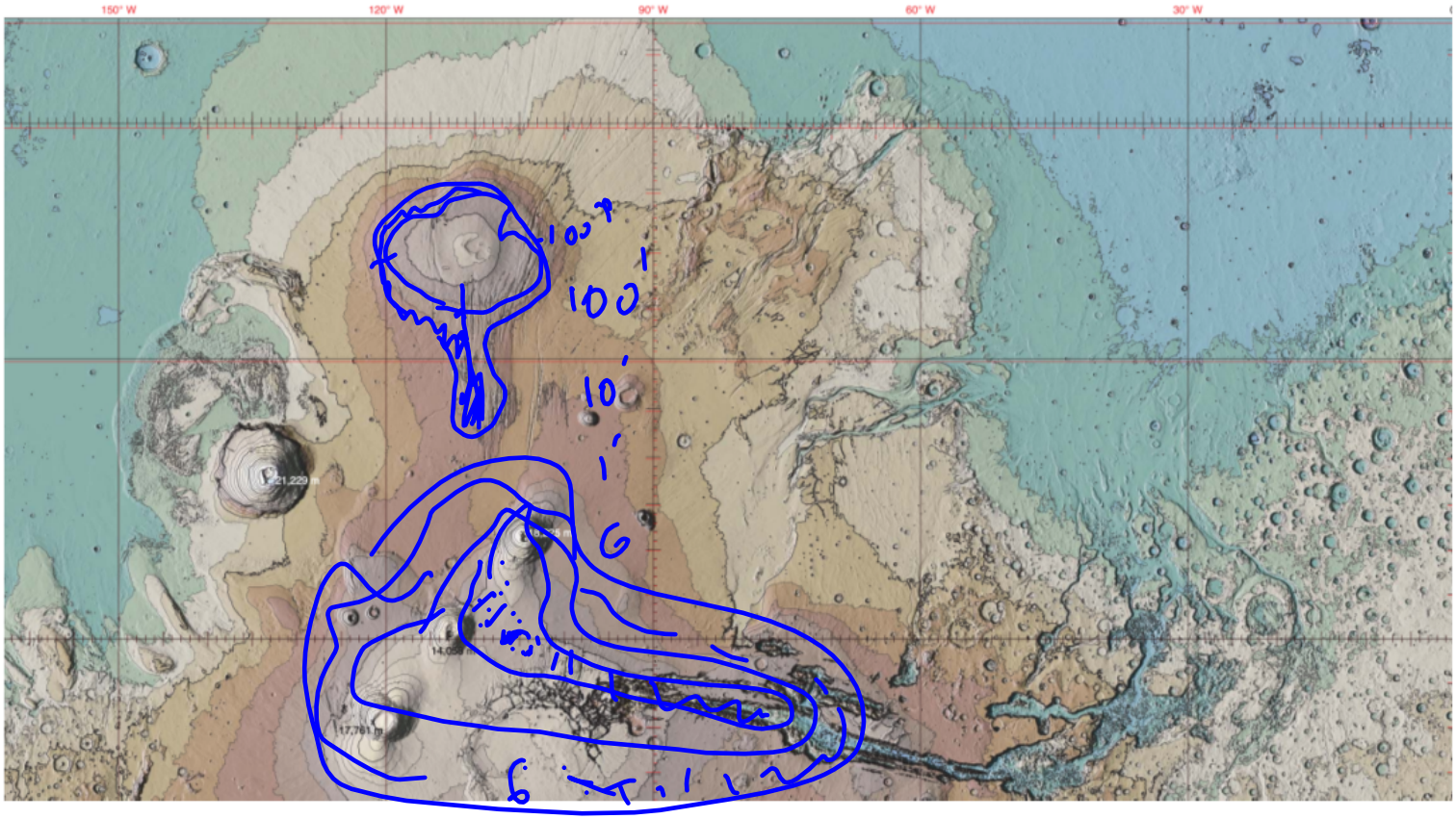


Image Credit: USGS

Alternatives

- IDA*: Iterative Deepening A*
- SMA*: Simplified Memory-Bounded A*
- RBS: Recursive best-first search.

RBFS

RBFS (Korf 1993)

- Linear-space heuristic search algorithm.
- Performs recursive search with a variable cutoff.
- Duplicates effort to save space.

RBFS: Operations

1. Called with 3 parameters:
   - Node n
   - Working Value V
   - Search Bound B

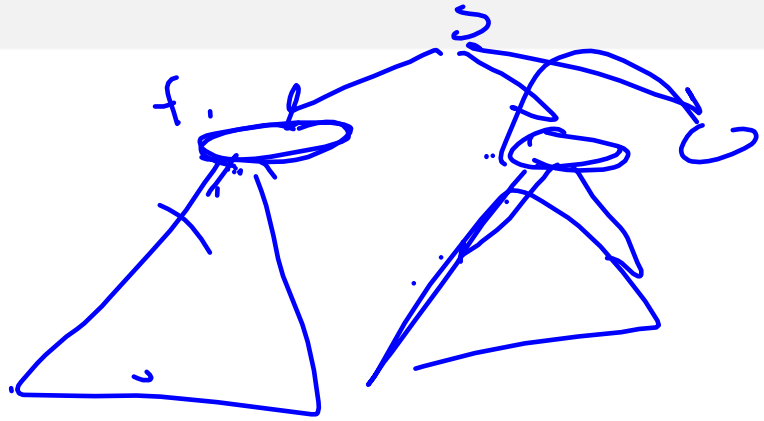
RBFS: Operations

1. Called with 3 parameters:
   - Node n
   - Working Value V
   - Search Bound B
2. If $f(n) > B$ then return $f(n)$
3. If $goalp(n)$ then success.
4. If $children(n) = \emptyset$ then ∞

RBFS: Operations

1. Called with 3 parameters:
   - Node n
   - Working Value V
   - Search Bound B
2. If $f(n) > B$ then return $f(n)$
3. If $goalp(n)$ then success.
4. If $children(n) = \emptyset$ then ∞
5. Create a priority queue of the children c_i:
   - 5.1 if $f(n) < V \wedge f(c_i) < V$ then $F_i = V$
   - 5.2 else $F_i = f(c_i)$
6. $F_{(|children(n)|+1)} = \infty$

RBFS: Operations



1. Called with 3 parameters:
   - Node n
   - Working Value V
   - Search Bound B
2. If $f(n) > B$ then return $f(n)$
3. If $goalp(n)$ then success.
4. If $children(n) = \emptyset$ then ∞
5. Create a priority queue of the children c_i:
   - **5.1** if $f(n) < V \wedge f(c_i) < V$ then $F_i = V$
   - **5.2** else $F_i = f(c_i)$
6. $F_{(|children(n)|+1)} = \infty$
7. while $F_0 <= B$
   - **7.1** Calculate a new F_0 as $RBFS(c_0, F_0, min(F_1, B))$
   - **7.2** Update the queue.
8. Return F_0


Korf, Richard E. (1993). "Linear-Space Best-First Search". In: *Artif. Intell.* 62.1, pp. 41–78. DOI: 10.1016/0004-3702(93)90045-D. URL: https://doi.org/10.1016/0004-3702(93)90045-D.