A* IN PROTEIN (RE)DESIGN

Presented by Nathan Guerin

CS/CBB 590

OUTLINE

1. Motivation
2. A* Algorithm
3. Properties of A*

MOTIVATION

A TYPICAL ROTAMER LIBRARY

Type	# Rotamers
Arginine	34
Lysine	27
Methionine	13
Glutamate	8
...	?

Suppose we have a decapeptide composed entirely of Arginine.

Q: How many possible conformations are there?

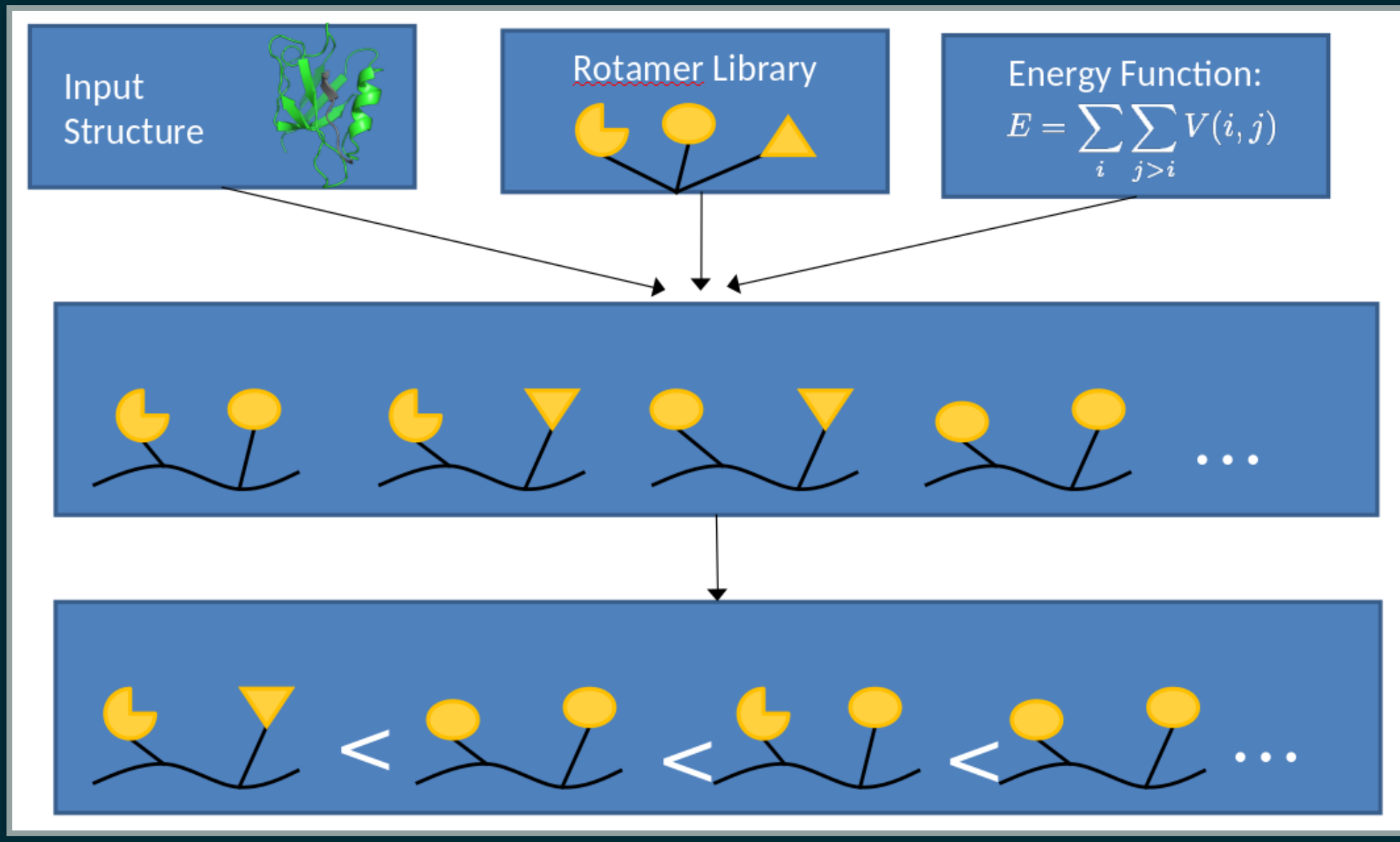
A: $37^{10} = 2.0644 \times 10^{15}$ conformations

TIME ANALYSIS

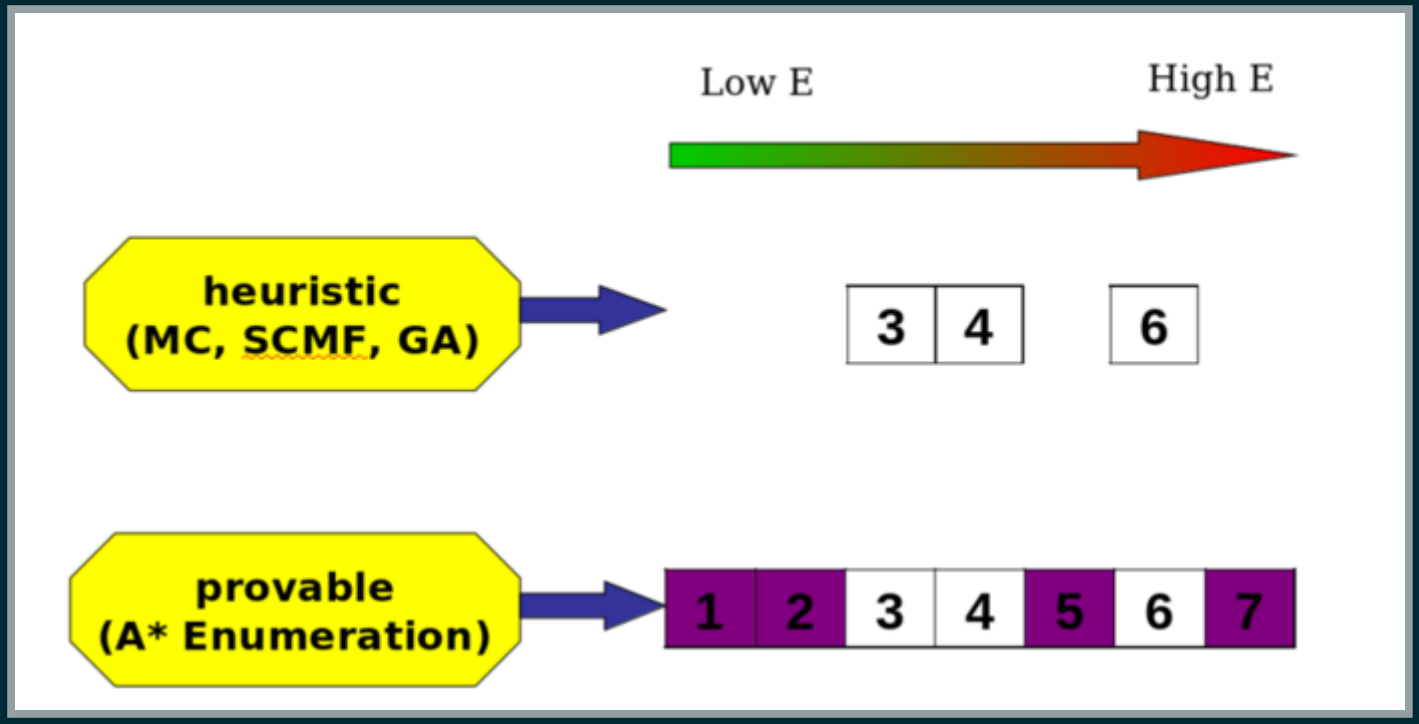
- Recall: 2.0644×10^{15} conformations
- Assume enumeration is instantaneous, energy calculation takes 10 ms
- Brute force takes ~650 000 years on a single processor to find GMEC

Obviously this isn't going to work...

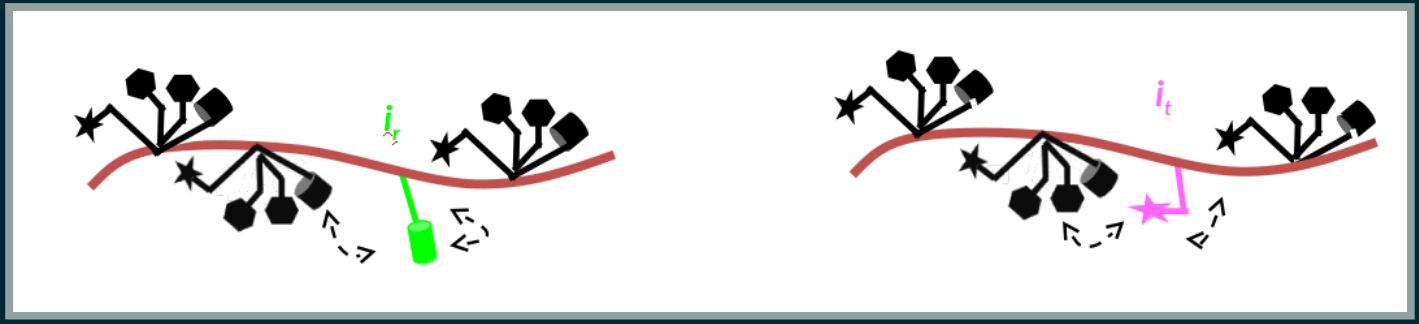
HOW IS THIS ACTUALLY DONE?

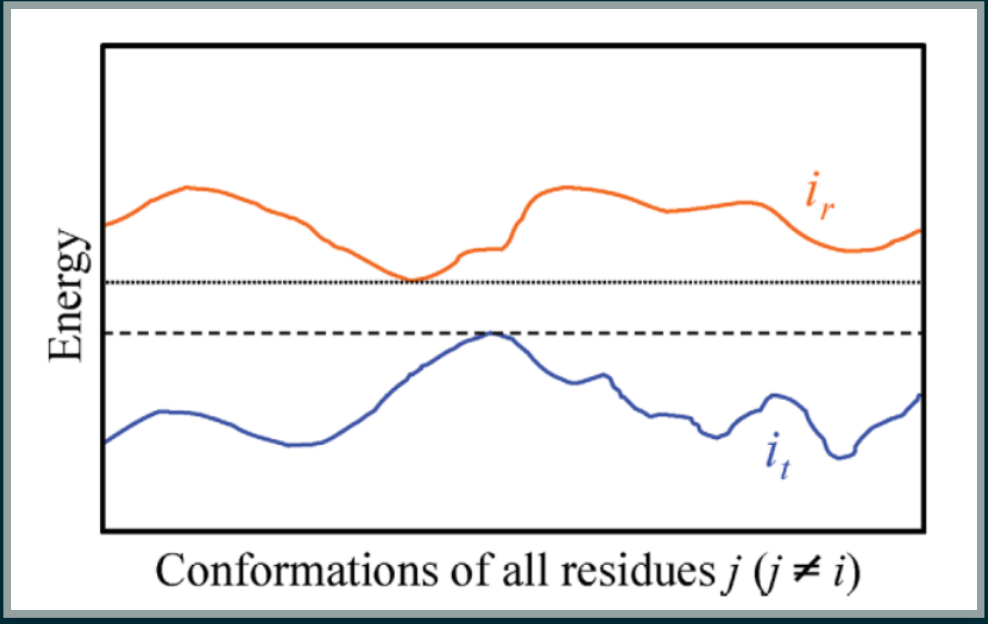


BENEFITS OF PROVABLE ALGORITHMS



DEAD-END ELIMINATION (DEE)





See Georgiev et al, 2008 for more information

THE DEE CRITERION

Given:

$$E_T = E_{t'} + \sum_i E(i_r) + \sum_i \sum_{j>i} E(i_r, j_s)$$

We can observe:

$$\forall i, j (i \neq j), \max_{s \in R_j} E(i_t, j_s) \geq E(i_t, j_g)$$

$$\min_{s \in R_j} E(i_g, j_s) \leq E(i_g, j_g)$$

DEE CRITERION: ENERGY WINDOW

Thus if any i_r satisfies this inequality:

$$E(i_r) + \sum_{j \neq i} \min_s E(i_r, j_s) >$$

$$E(i_t) + \sum_{j \neq i} \max_s E(i_t, j_s) + E_w$$

It can be pruned.

CHOICES REMAIN. WE STILL NEED TO:

- Find GMEC
- Find ordered list of low-energy confs (for free energy approximation)
- Finish quicker than brute-force enumeration

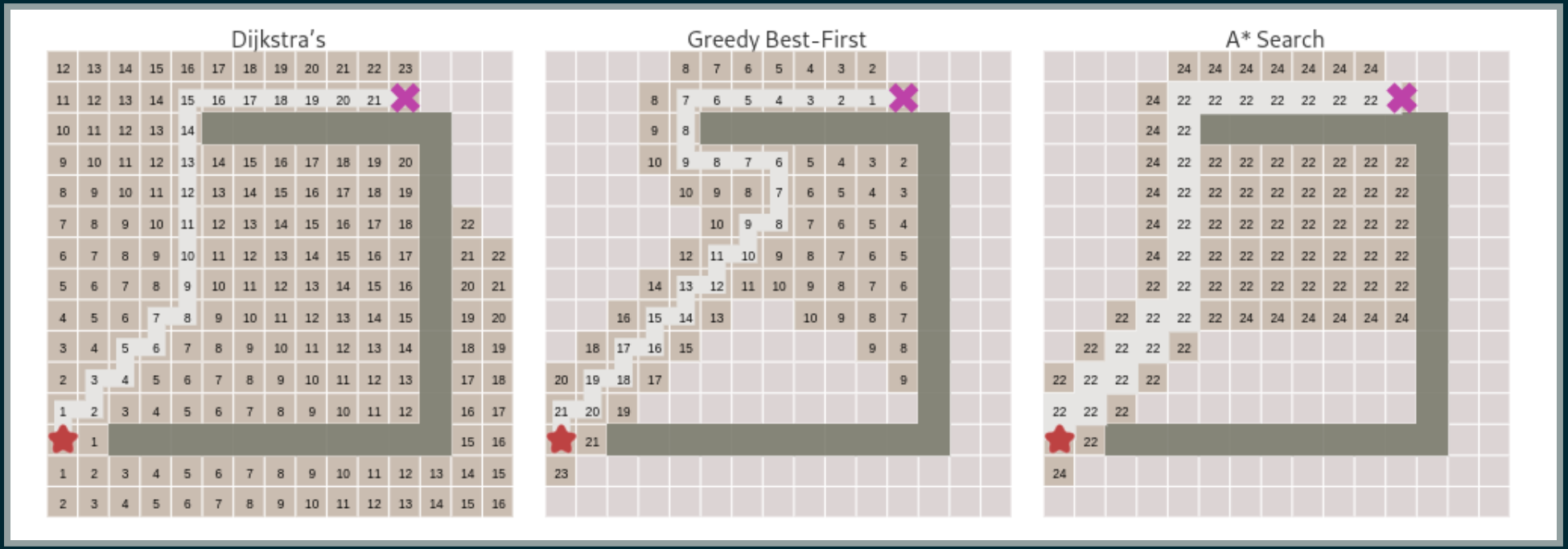
THE A* ALGORITHM

A BRIEF HISTORY

- Originally developed by Hart, Nilsson, and Raphael in 1968
- Leach and Lemon applied to discrete rotamer problem in 1998
- Georgiev et al. developed minimization-aware A* in 2008

ABOUT A*

- Fundamentally a graph-traversal problem
- Finds lowest-cost path from point A to point B



Source

THE PATH AND COST IN PROTEIN DESIGN

- The path is the amino acid sequence. The target is reached when all residues are assigned an amino acid and rotamer
- The cost is the energy of the conformation, calculated by an energy function
- Different amino acid and rotamer choices incur varying costs

A* SEARCH: DEFINITIONS

$$f(x) = g(x) + h(x)$$

- x is a path, with nodes x_{start} and x_{end}
- $f(x)$ is a lower bound on the cost of the complete path
- $g(x)$ is the known cost from x_{start} to x_n
- $h(x)$ a heuristic estimating the remaining cost from x_n to x_{end}

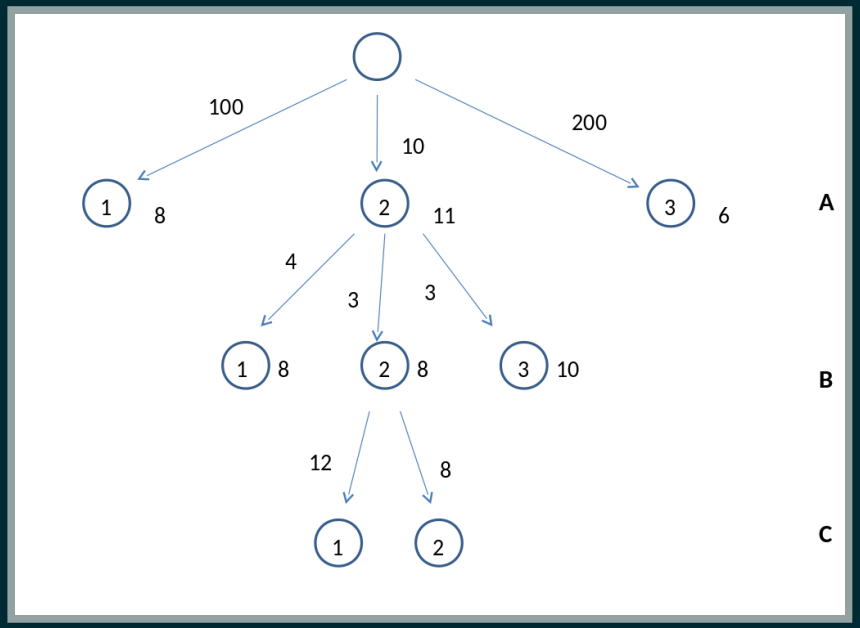
A* SEARCH: APPLIED TO PROTEINS

$$f(x) = g(x) + h(x)$$

- x is a backbone template, with residues x_{start} to x_{end}
- $f(x)$ is a lower bound energy of the protein
- $g(x)$ is the known energy contributions from x_{start} to x_n
- $h(x)$ a heuristic estimating the energy contributions of x_n to x_{end}

A* SEARCH: THE PARTICULARS

The A* search for protein design uses a tree.



Each level corresponds to a residue index; each node to an amino acid+rotamer at that index. The numbers next to the nodes estimated costs $h(x)$ and the numbers next to the edges the actual costs $g(x)$.

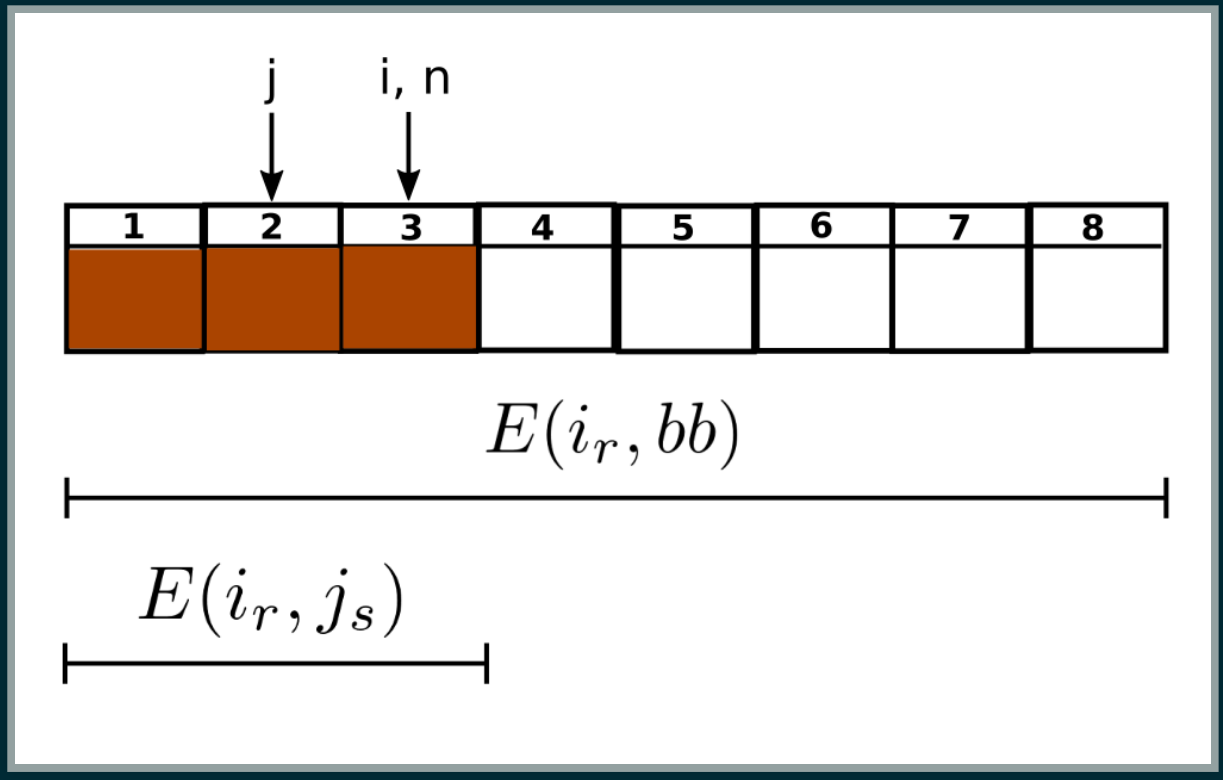
A* SEARCH IN RELATION TO OTHER ALGORITHMS

$$f(x) = g(x) + h(x)$$

- When $h(x)$ is 0, we have Dijkstra's algorithm
- When $g(x)$ is 0, we have greedy best-first search
- See more here

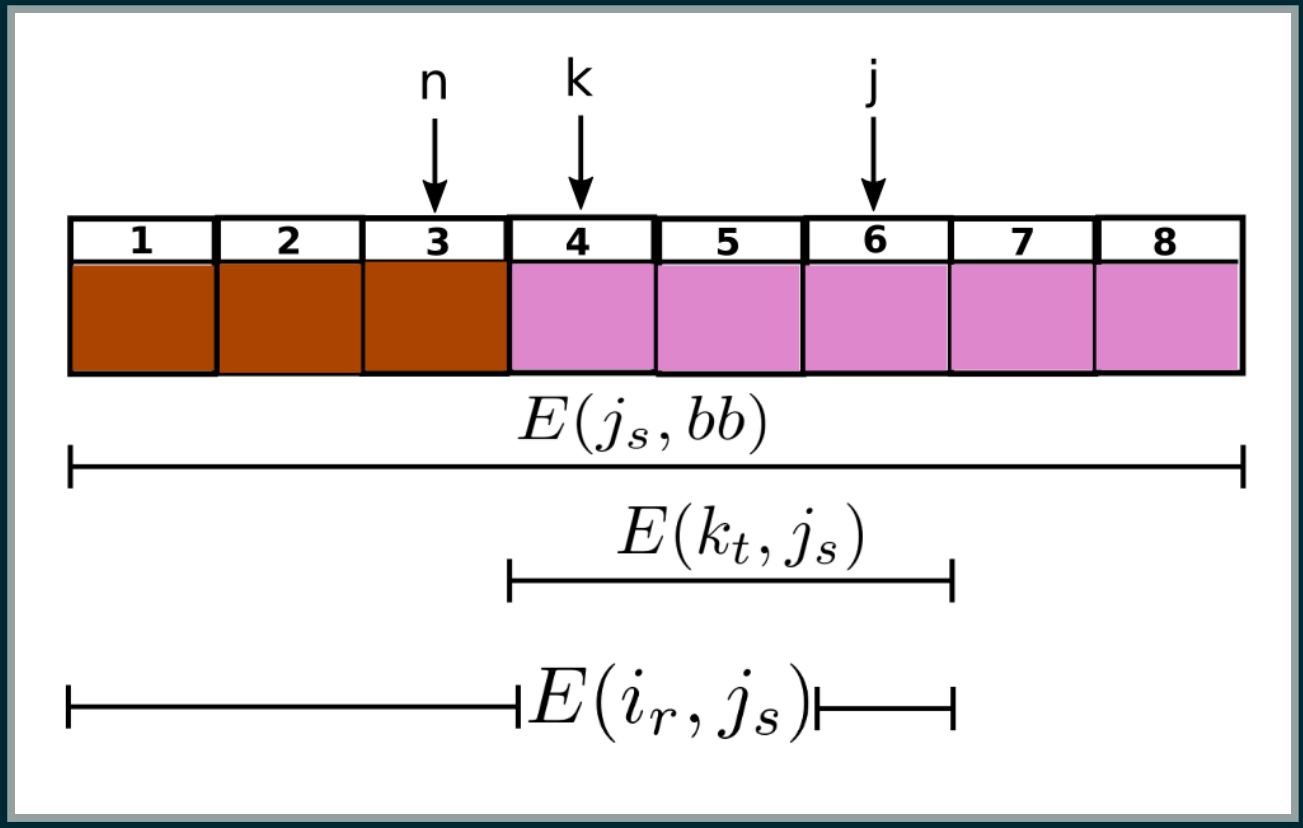
VALUE OF TAKEN PATH

$$g(x_n) = \sum_{i=1}^{n} [E(i_r) + E(i_r, bb) + \sum_{j=1}^{i-1} E(i_r, j_s)]$$



ESTIMATION OF HEURISTIC

$$h(x_n) = \sum_{j=n+1}^{N} \min_s [E(j_s) + E(j_s, bb) + \sum_{i=1}^{n} E(i_r, j_s) + \sum_{k=n+1}^{j-1} \min_t E(k_t, j_s)]$$



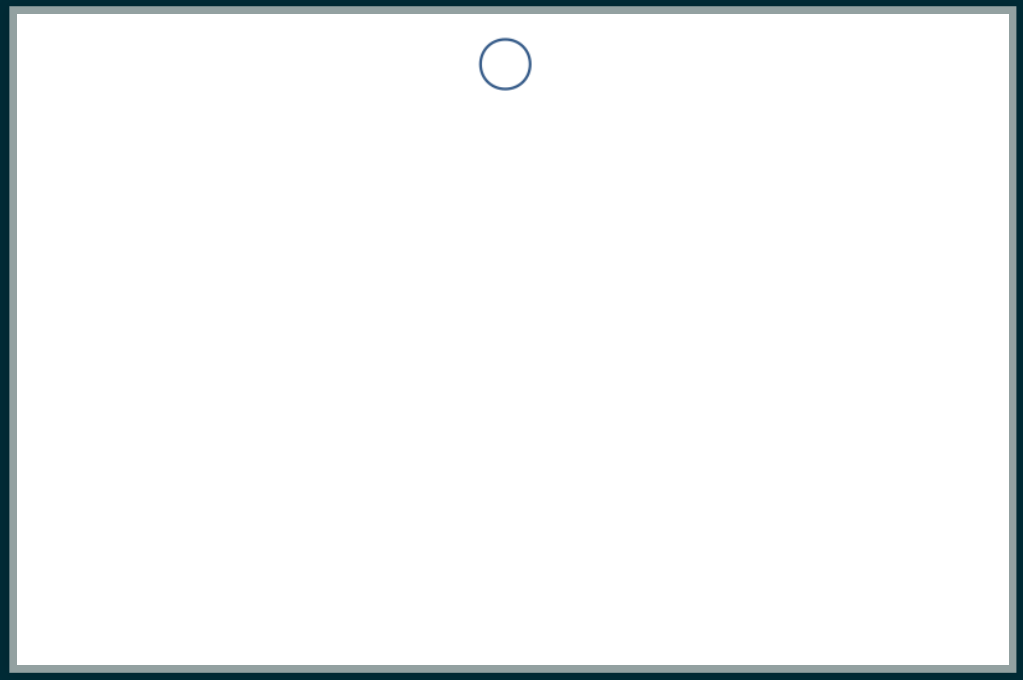
KEEP A PRIORITY QUEUE

- The search always expands nodes with the minimum $f(x)$. It maintains a list of expanded nodes
- Thus, it expands nodes that it anticipates will lead to the best energy
- See binary heap implementation of priority queue here

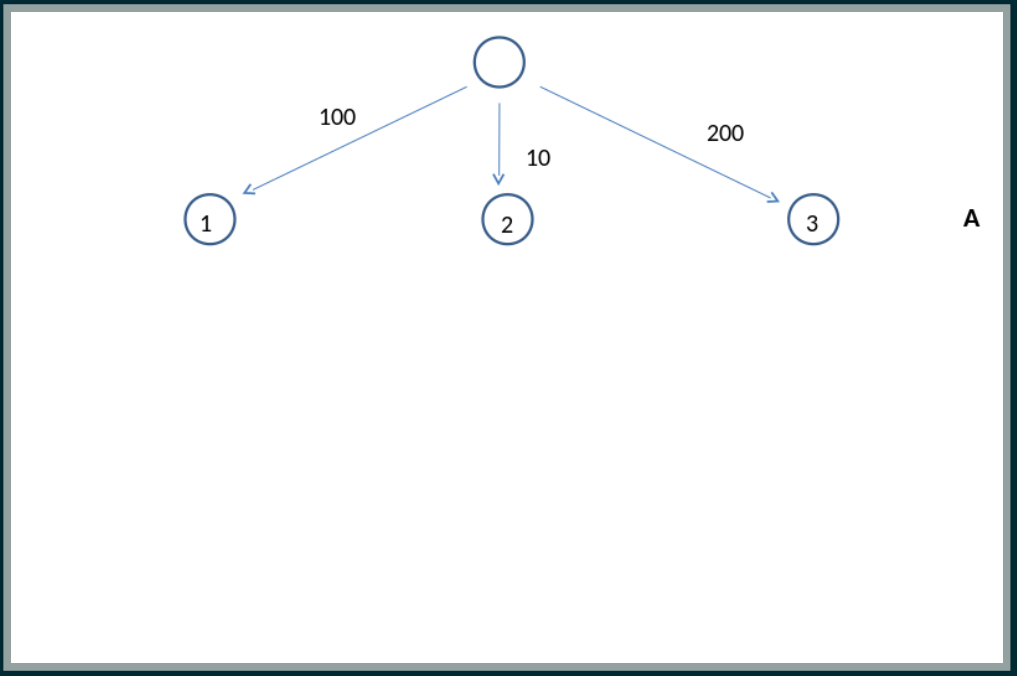
EXAMPLE: LET'S DESIGN A TRIPEPTIDE

- After DEE pruning:
  - Residue A - 3 Rotamers
  - Residue B - 3 Rotamers
  - Residue C - 2 Rotamers
- Assume values of $g(x)$ and $h(x)$ are given at each node
- (From Leach and Lemon, 1998)

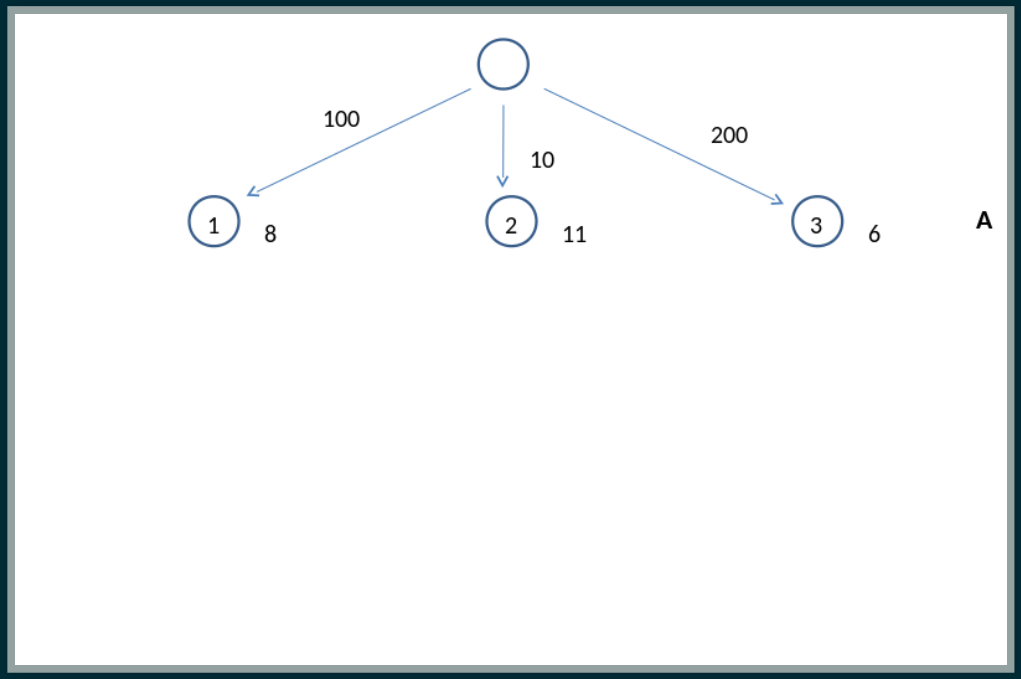
EXAMPLE

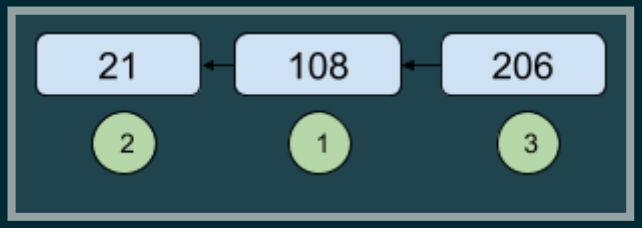


EXAMPLE

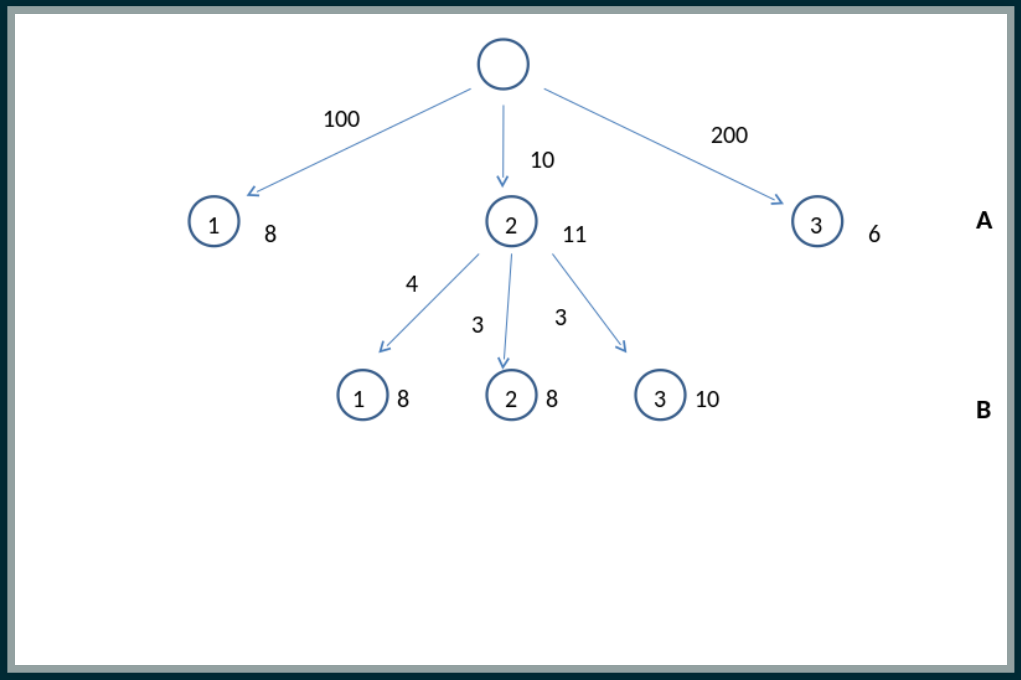


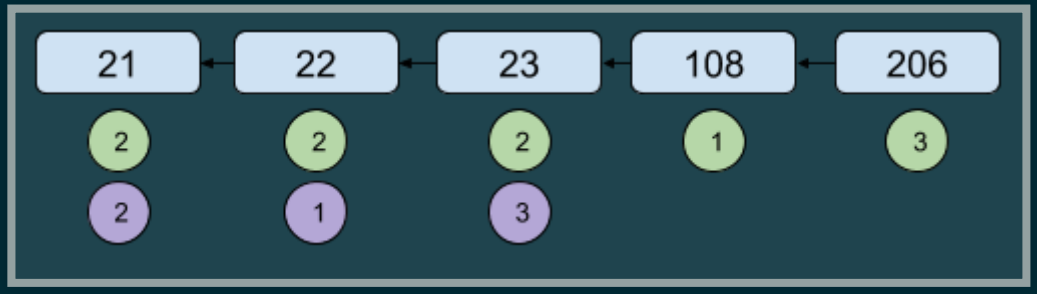
EXAMPLE



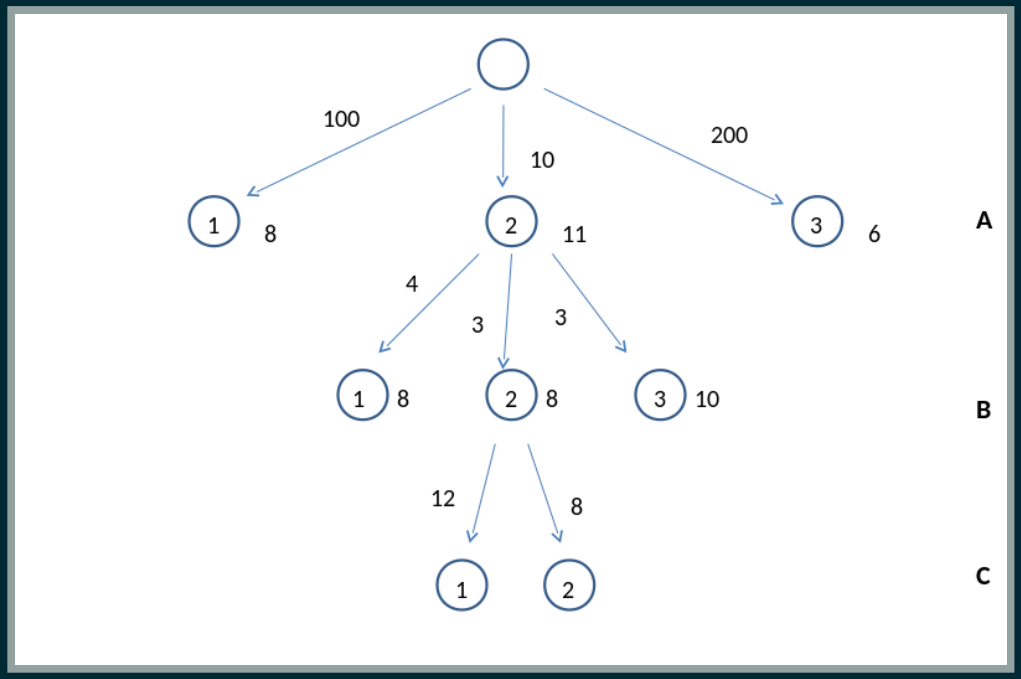


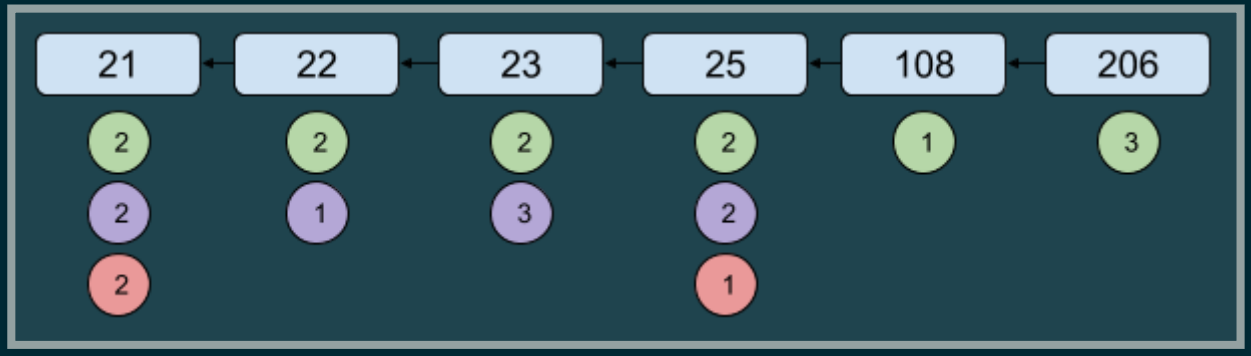
EXAMPLE



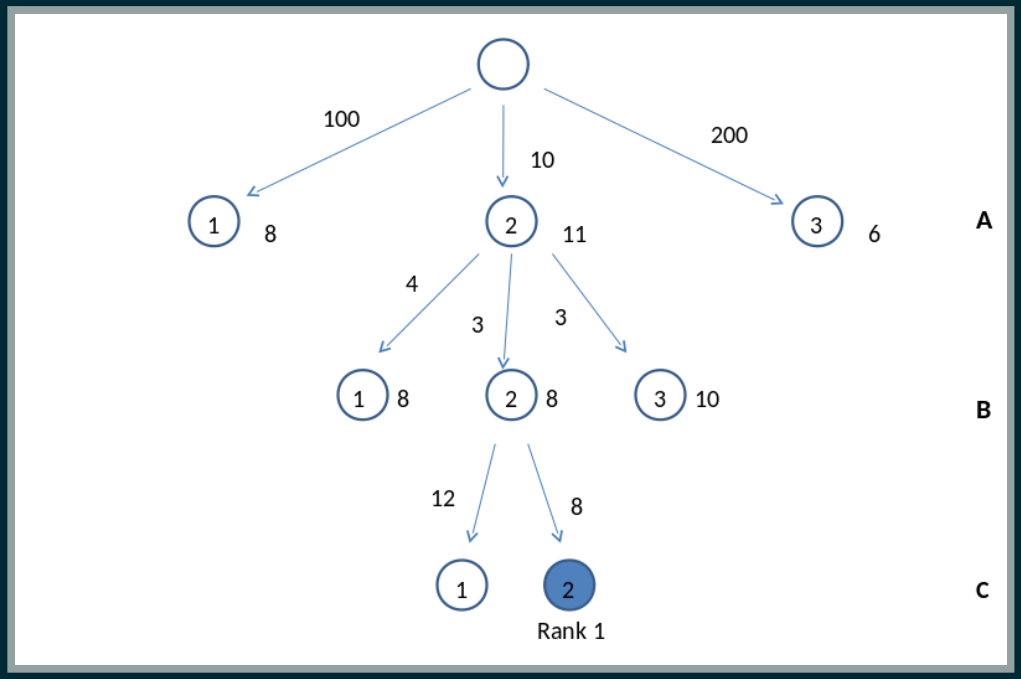


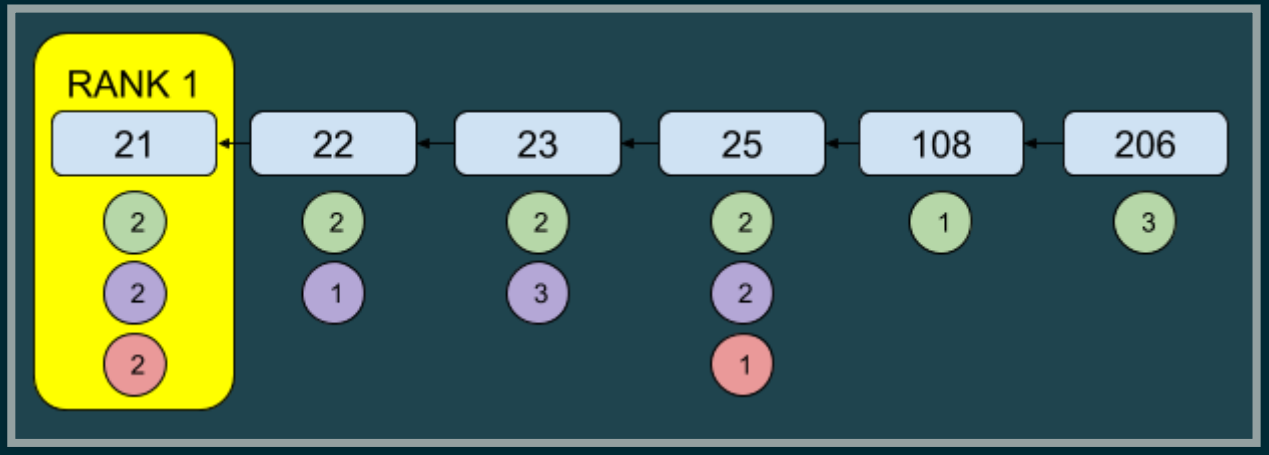
EXAMPLE


100
10
200
1
8
2
11
3
6
A
4
3
3
1
8
2
8
3
10
B
12
8
1
2
C

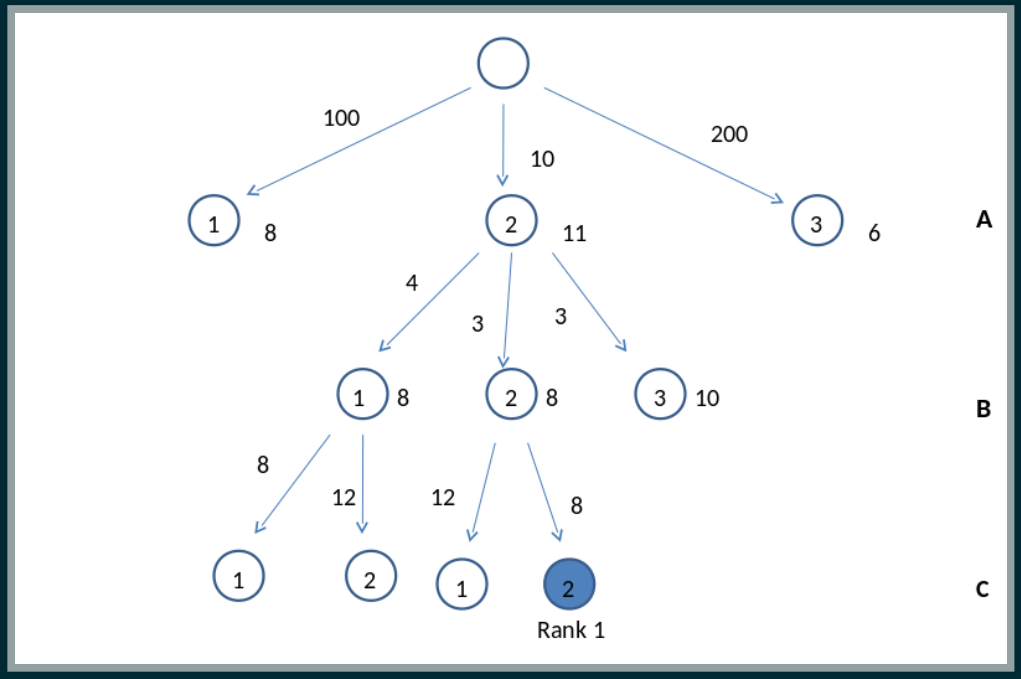

21
22
23
25
108
206
2
2
2
2
1
3
2
1
3
2
1
2

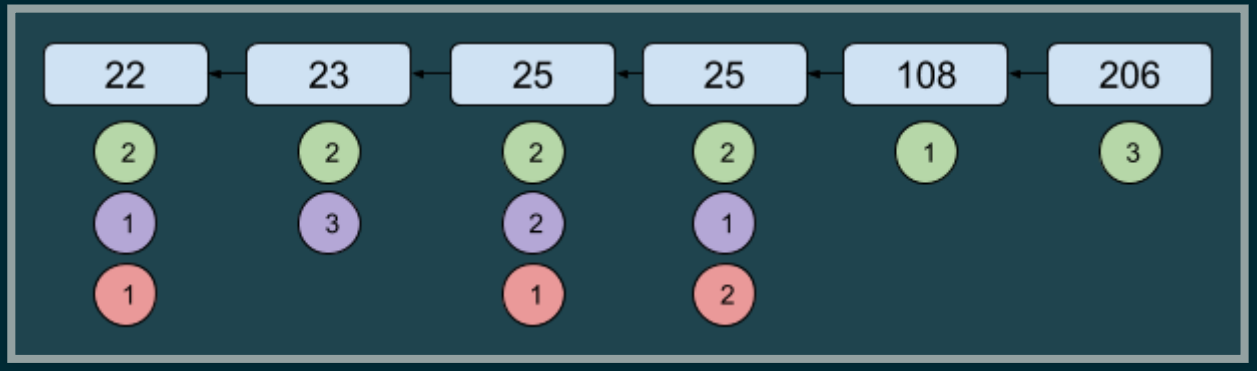
EXAMPLE


100
10
200
1
8
2
11
3
6
A
4
3
3
1
8
2
8
3
10
B
12
8
1
2
Rank 1
C

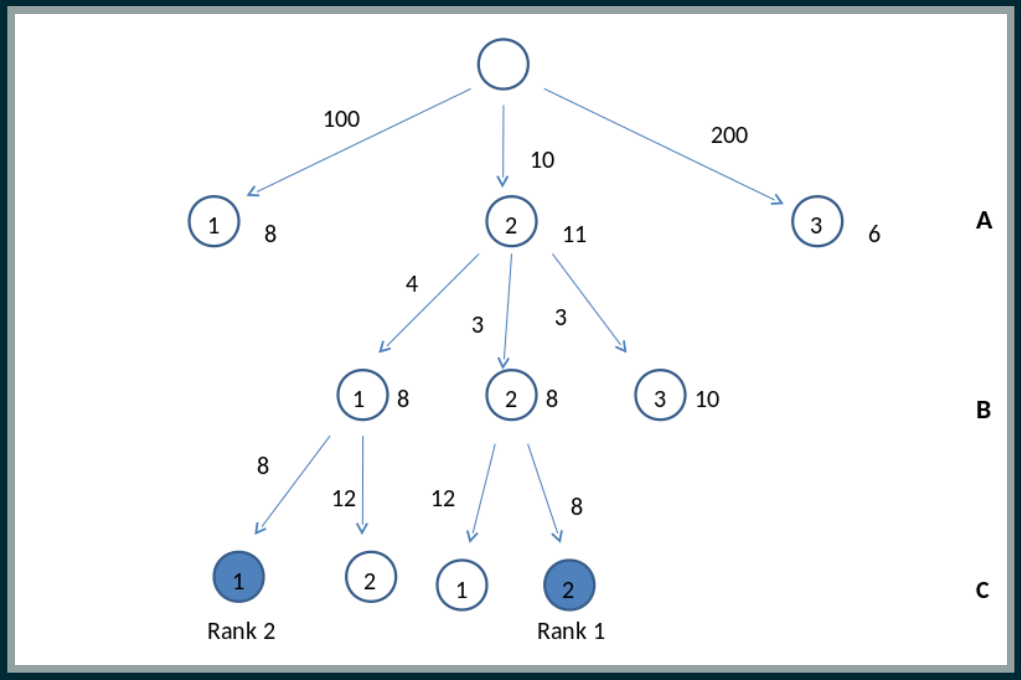

RANK 1
21
22
23
25
108
206
2
2
2
2
1
2
3
2
2
1
1
3

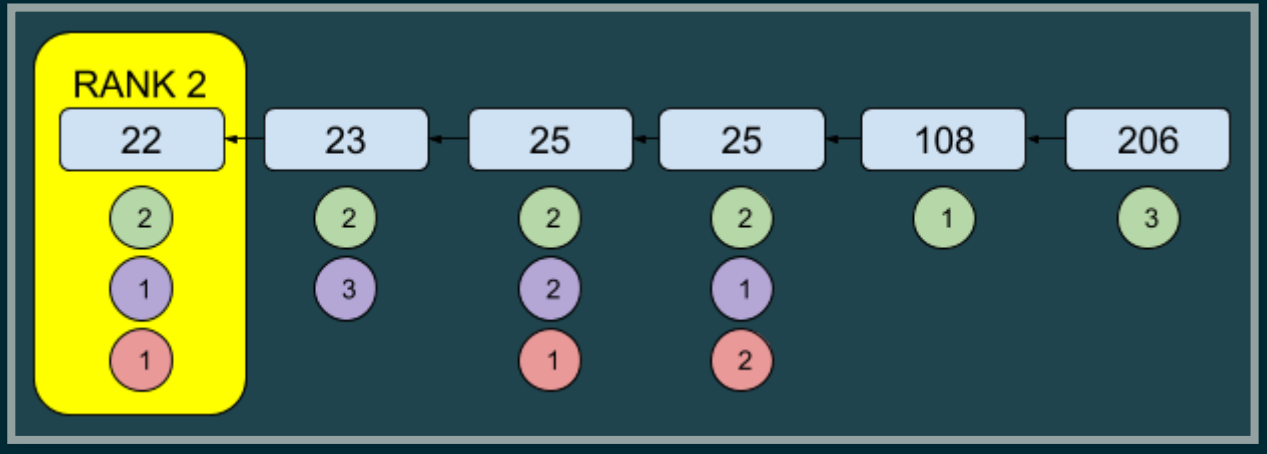
EXAMPLE


100
10
200
1
8
2
11
3
6
A
4
3
3
1
8
2
8
3
10
B
8
12
12
8
1
2
1
2
C
Rank 1


22
23
25
25
108
206
2
1
1
2
3
2
1
2
3
1
2
1
2

EXAMPLE


100
10
200
1
8
2
11
3
6
A
4
3
3
1
8
2
8
3
10
B
8
12
12
8
1
2
1
2
Rank 2
Rank 1
C


RANK 2
22
23
25
25
108
206
2
1
1
2
3
2
1
2
1
2
1
3

A* ALGORITHM - PROPERTIES

ADMISSIBILITY

A graph search algorithm is said to be admissible if it finds the optimal path p from s (start) to t (target) if a path exists.

- Dijkstra's algorithm is admissible
- Greedy best-first search is not

CLAIM: A* IS ADMISSIBLE

1. A* terminates
2. A* finds the GMEC (and additional conformations in order of increasing energy)

A* TERMINATES

Since we're dealing with proteins, assume a finite number of residues

1. Graph is an acyclic, directed tree with depth equal to number of residues
2. Thus, there are a finite number of paths from root to leaves
3. Each root-to-leaf conformation is removed without adding new items to priority queue
4. Thus, eventually the priority queue will run out of items and the algorithm will terminate

A* FINDS OPTIMAL PATH: DEFINITIONS

- $f^*(s)$ - optimal path from s to t, $\{n_s, n_2, \ldots, n_t\}$
- $g^*(n)$ - optimal path from s to n
- $h^*(n)$ - optimal path from n to t

A* FINDS OPTIMAL PATH

1. Assume we are on node n', and n' is on the optimal path $s \ldots t$. Then $f(n') = g(n') + h(n')$.
2. Since n' on optimal path, $g(n') = g^*(n')$
3. Moreover, $h(n') \leq h^*(n')$
4. Therefore, $f(n') \leq g^*(n') + h^*(n') = f^*(n')$
5. Since optimal path passes through n', $f^*(n') = f^*(s)$ and $f(n') \leq f^*(s)$
6. Implication: prior to reaching target, there will always be a n' in priority queue with $f(n') \leq f^*(s)$

A* FINDS OPTIMAL PATH (CONTINUED)

Proof by contradiction: Assume that we've completely assigned rotamers to a conformation t and $f(t) > f^*(s)$. But since $f(n') \leq f^*(s)$, $f(n') < f(t)$. Since by the previous result we know $f(n')$ is in the priority queue, then $A*$ would have chosen $f(n')$ before $f(t)$, leading to $f^*(s)$ before $f(t)$. This contradicts the assumption that the algorithm would complete $f(t)$ first.

IS A* ADMISSIBLE?

- Recall: Admissibility is termination and optimality
- Shown A^* terminates
- Shown A^* finds optimal path
- Thus, $A*$ is admissible

OTHER PROPERTIES WORTH MENTIONING

- A more powerful, yet admissible, heuristic expands fewer nodes
- If a heuristic satisfies a monotone restriction $h(n_i) \leq h(n_j) + c(n_i, n_j)$, then the start to any expanded node is the optimal path to that node.
- Proofs in Nilsson's *Principles of Artificial Intelligence*

CREDITS

- Swati Jain's presentation from Algorithms for Drug Design for many images
- Nils Nilsson's "Principles of Artifical Intelligence" for proofs
- Amit Patel's website Red Blog Games

QUESTIONS?