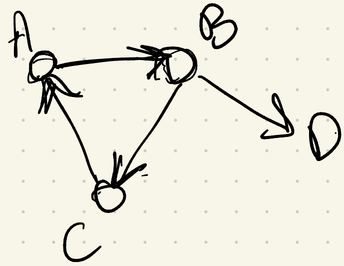
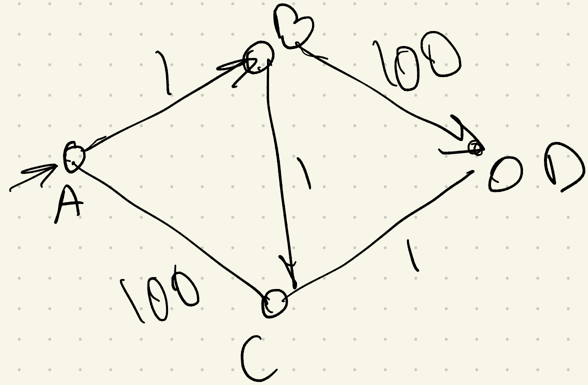
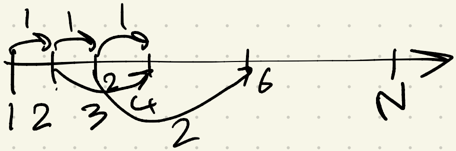
Search II:





Explored	Frontier		Unexplored
A	A	0	
B	B	1	
C	C	2	
	D	3	



1
1
1
2
6
4
2
1 2
3
N

$S_0 \xrightarrow{\text{cost}'(a_1)+\text{cost}'(a_2)+\cdots} S$

$+h(S)-h(S_0)$

General framework

Definition: relaxed search problem

A **relaxation** P_{rel} of a search problem P has costs that satisfy:

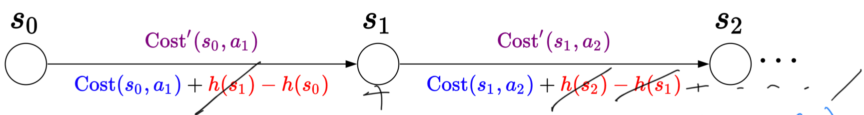
$$\text{Cost}_{\text{rel}}(s,a) \le \text{Cost}(s,a).$$

Definition: relaxed heuristic

Given a relaxed search problem P_{rel}, define the **relaxed heuristic** $h(s) = \text{FutureCost}_{\text{rel}}(s)$, the minimum cost from s to an end state using $\text{Cost}_{\text{rel}}(s,a)$.

Proof of A* correctness

- Consider any path $[s_0, a_1, s_1, \ldots, a_L, s_L]$:



$\text{cost}(s_0, a_1) + \text{cost}(s_1, a_2) + \ldots + \text{cost}(s_{n-1}, a_n)$

$\text{cost}'(s_0, a_1) + \ldots + \text{cost}'(s_{n-1}, a_n)$

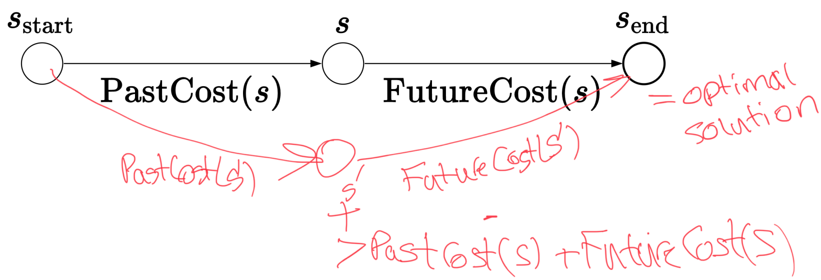
$= \text{cost}(s_0, a_1) + \ldots + \text{cost}(s_{n-1}, a_n)$

$+ h(\cancel{s_n}) - h(s_0)$

0

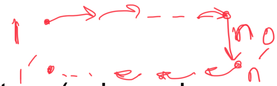
Exploring states

UCS: explore states in order of $\mathrm{PastCost}(s)$



answer in chat

Question

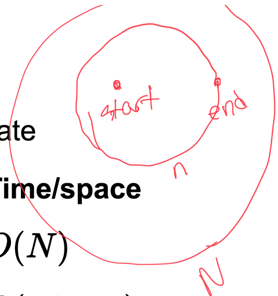


Suppose we want to travel from city 1 to city n (going only forward) and back to city 1 (only going backward). It costs $c_{ij} \geq 0$ to go from i to j. Which of the following algorithms can be used to find the minimum cost path (select all that apply)?

- depth-first search
- breadth-first search
- dynamic programming
- uniform cost search

activate deactivate reset report

DP versus UCS



N total states, n of which are closer than end state

Algorithm	**Cycles?**	**Action costs**	**Time/space**
DP	no	any	$O(N)$
UCS	yes	≥ 0	$O(n \log n)$

Note: UCS potentially explores fewer states, but requires more overhead to maintain the priority queue

Note: assume number of actions per state is constant (independent of n and N)

Analysis of uniform cost search

Theorem: correctness

When a state s is popped from the frontier and moved to explored, its priority is $\mathrm{PastCost}(s)$, the minimum cost to s.

Proof: