COMP 472 Artificial Intelligence
State Space Search
Informed Search
More on Heuristics & Summary

part 3

video 6

- Russell & Norvig – Section 3.5.2

Today

1. State Space Representation
2. State Space Search
    a) Overview
    b) Uninformed search
        1. Breadth-first and Depth-first
        2. Depth-limited Search
        3. Iterative Deepening
        4. Uniform Cost
    c) Informed search
        1. Intro to Heuristics
        2. Hill climbing
        3. Greedy Best-First Search
        4. Algorithms A & A*
        5. More on Heuristics

    d) Summary

Evaluating Heuristics

$\forall\ h(n) = 0$

1. Admissibility:
   - "optimistic"
   - $h(n)$ never overestimates the actual cost of reaching the goal $\forall n\ h(n) \le h^*(n)$
   - guarantees to find the lowest cost solution path to the goal (if it exists) A*

2. Monotonicity: 比上面那个还强
   - "local admissibility" 上面那个只需要node本身的estimate小于actual cost,这个要求这个Node以及它的children的estimate都小于actual
   - guarantees to find the lowest cost path to each state n visited (i.e. popped from OPEN) 确保pop到close里面的state,必然是这个state的lowest cost

not informed

3. Informedness: 与上面两个无关，用来评估heuristic的好坏
   - measure for the "quality" of a heuristic
   - the more informed, the less backtracking, the shorter the search path 越Informed,backtracking越少，search path越短

Admissibility

- A heuristic is admissible if it never overestimates the cost of reaching the goal
  - i.e.:
    - h(n) ≤ h*(n) for all n
  - hence
    - h(goal) = h*(goal) = 0
    - h(n) = ∞ if we cannot reach the goal from *n*
- Algorithm A that uses an admissible heuristic
  - is called algorithm A*
  - guarantees to find the lowest cost solution path to the goal (if it exists) 可以确保如果我们找到解的话，从root到这个解的path最短
  - note: does not guarantee to find the lowest cost search path
  - e.g.: uniform cost is admissible -- it uses f(n) = g(n) + 0

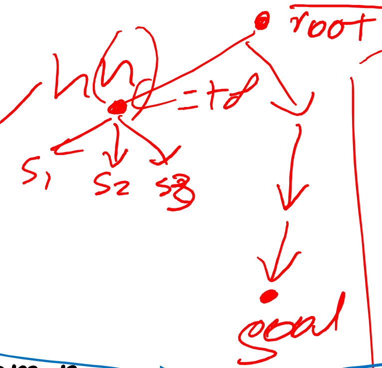
但不能保证search path最短
因为h(n)和h*n不等

actual

h*

guess

actual



if h(n) = h*(n) ∀n
guess
actual
h(n)一直等于actual cost是最完美的

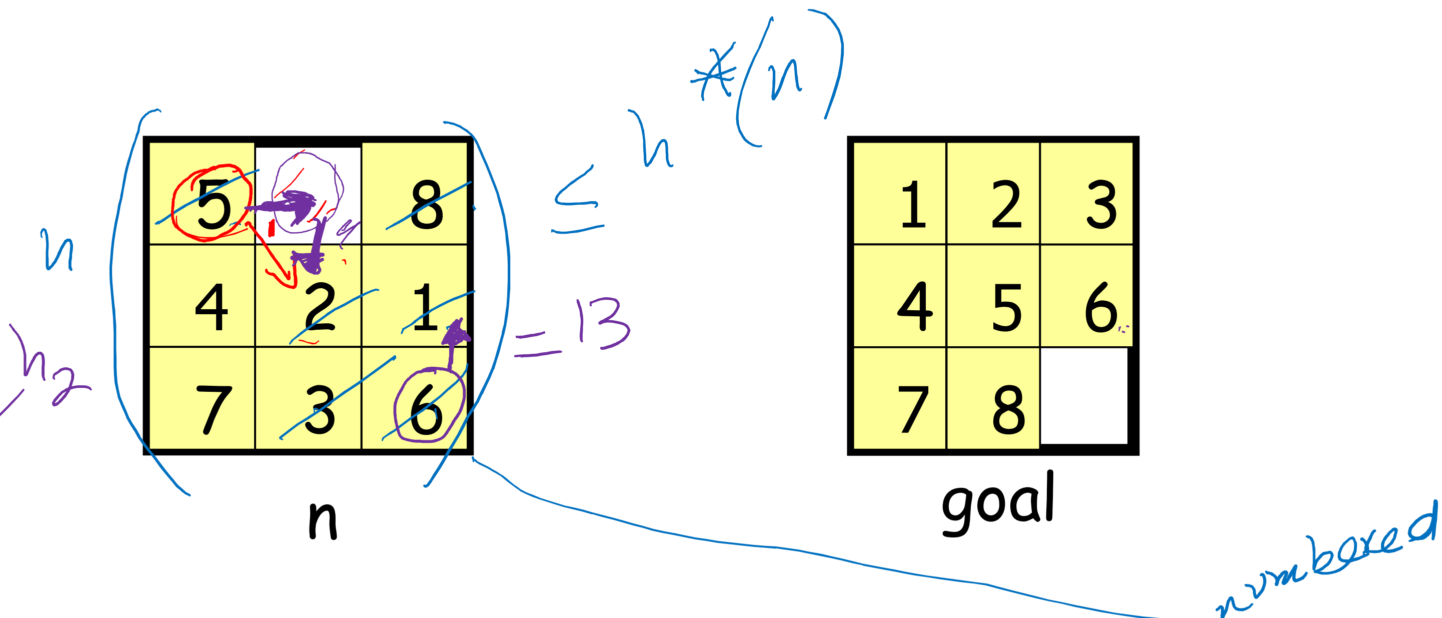
h(n)=0是uninformed，我们要大量backtracking，但是还是admissible的

if h(n) = 0 ∀n

i.e you can backtrack

it is admissible but uninformed

Example: 8-Puzzle

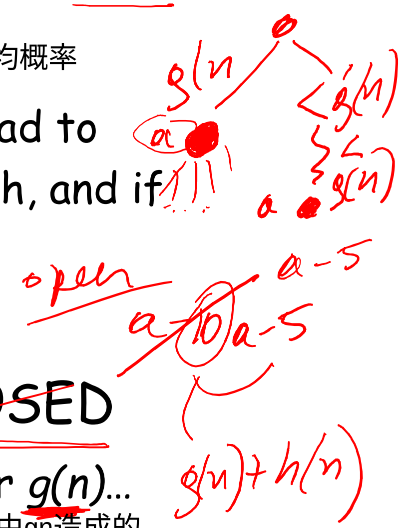


- h1(n) = Hamming distance = number of misplaced tiles = 6
--> admissible 一步到终点，显然没有over estimation
- h2(n) = Manhattan distance = 13
--> admissible 假设不会卡死，5移动到goal需要2， 仍然是admissible，且更接近actual cost

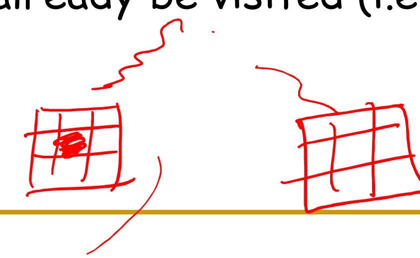
Problem with Admissibility

意思就是在你没达到终点前，可能有一个点A，他当前消费是10，我们把它当成暂时最优解，但是我们过一会儿找到了一个消费是3的A，哪怕A-10已经加入CLOSED-LIST了，为了完成最优解我们也要用A-3替换他

- Admissible heuristics may temporarily reach non-goal states along a suboptimal path 次最优的
  经典例子就是uniform-cost，他是admissible,每个均概率
  上层的必然比下层的先循环
  - remember with uniform-cost… when we expanded a node, we had to check if it was already in the OPEN list with a higher path path, and if so, we would replace it with the current path cost/parent info



- With A*, if we have a node *n* in OPEN or even in CLOSED
  - We may later find *n* again, but with a lower *f(n)* (due to a lower *g(n)*)… 是由gn造成的 the *h(n)* will, by definition be the same). 因为同样的state hn相同
  - So to ensure that the solution path has the lowest cost,
    - We may need to update the cost/parent info of node *n* in OPEN or even put *n* back in OPEN even if it has already be visited (i.e. in CLOSED)… expensive work…



Admissibility and A* Search

- Admissibility: $\forall n \; h(n) \leq h^*(n)$
  - To guarantee to find the lowest cost solution path, when we generate a successors s: i.e. when node n is popped from OPEN 当node a 被pop出来的时候

如果想确保lowest cost就是必要的

expensive but necessary if you want to guarantee lowest cost solution

1. IF s is already in CLOSED
   IF s in CLOSED has a higher f-value due to a higher g-value i.e g(s)
   THEN place s and its new lower f-value in OPEN ! 我们需要吧更低的值重置回OPEN
   // we found a lower cost path to s, but we had already expanded s…
   // to guarantee the lowest cost solution path, we need to put s back in OPEN and re-visit it again
   ELSE ignore s
2. ELSE IF s is already in OPEN
   IF s in OPEN has a higher f-value
   THEN replace the old s in OPEN with the new lower f-value s 已经在OPEN里，用新open替换
   // we found a lower cost path to s, and we had not expanded s yet
   // to guarantee the lowest cost solution path, we need to replace the old s in OPEN with the new
   // lower-cost s
   ELSE ignore s
3. ELSE insert s in OPEN 普通的就直接插入OPEN
   // as usual

Monotonicity (aka consistent)

- Admissibility: 不能确保每一个expanded的node n都是我们要的lowest cost path中的一员
  - does not guarantee that every node *n* that is expanded (i.e. for which we generate the successors *s*) will have been found via the lowest cost, the first time we expand it
- Monotonicity
  - guarantees that! 它的定义就是保证每一个open里Pop到close以后，必然就是最小的，
  - Stronger property than admissibility
- If a heuristic is monotonic
  - We are guaranteed that once a node is popped from the OPEN list, we have found the lowest cost path to it

    root
    n
    n
  - i.e we always find the lowest cost path to each node, the 1st time it is popped from OPEN!
  - So once a node is placed in the CLOSED list, if we encounter it again, we do not need to check that the 2nd encounter has a lower cost. We can just ignore it. (more efficient!) 如果再次遇到，就不用第二次check，可以直接忽视

Monotonicity vs Admissibility

预计的n的cost必须小于n到子Node的cost+预计的s cost

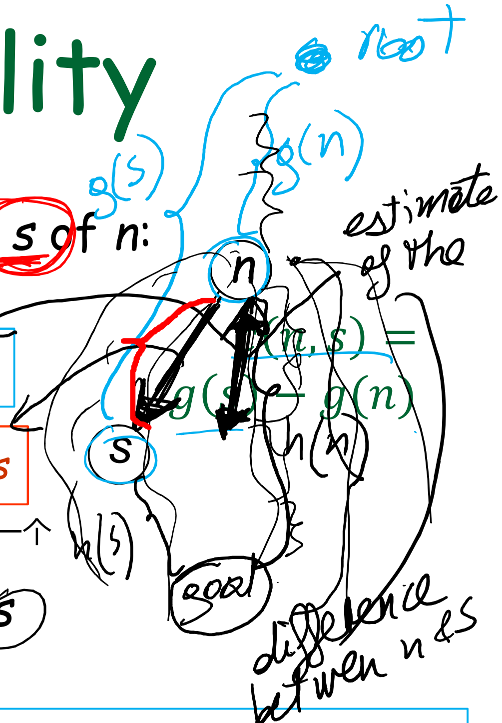
- *h* is monotonic if for every node *n* and every successor *s* of *n*:
  - $h(n) \le c(n,s) + h(s)\ \forall n,s$　h(n)<=c(n,s)+h(s)
  - $h(n) - h(s) \le c(n,s)$　Estimate of cost from n to s
  - $h(n) - h(s) \le g(s) - g(n)$　Actual of cost from n to s

第二个更好记

stronger

h不是这个点的用量，而是估计到goal的cost，hn-hs实际上与gs-gn指的是同一个东西，不过第一个是用h估计求出来的，第二个是通过已知G求出来的

- -> monotonic = *h(n)* is optimistic for all transitions *n-->s*
- *f(n)* is non-decreasing along any path



Estimate of cost from n to goal

- admissibility = *h(n)* only needs to be optimistic for *n-->goal*
  - $h(n) \le h^*(n)\ \forall n$　Actual of cost from n to goal
  - $h^*(n) = g(goal) - g(n) \rightarrow h(n) \le g(goal) - g(n)$　actual cost to goal- actual cost till now
  - $h(goal) = 0 \rightarrow h(n) - h(goal) \le g(goal) - g(n)$

- Every monotonic *h(n)* is admissible (but not vice-versa)

Monotonicity and A* Search

- Monotonicity
  - Guarantees to find the lowest cost solution path monotonicity必然是admissible
  - Guarantees to find the lowest cost path to every node, the first time we expand it. 第一次展开就能找到root到这个state的最短路径
  - --> no need to check the CLOSED list again!
  - So when we generate a successors s:

~~1. IF s is already in CLOSED~~

~~IF s in CLOSED has a higher f-value~~

~~THEN place s and its new lower f-value in OPEN !~~

~~*// we found a lower cost path to s, but we had already expanded s...*~~

~~*// to guarantee the lowest cost solution path, we need to put s back in OPEN and re-visit it again*~~

~~ELSE ignore s~~

2. ELSE IF s is already in OPEN 因此第一个IF不用了，

IF s in OPEN has a higher f-value 只有OPEN之间仍然需要比较 //不代表第一个找到的open就是最优解，第一个pop出Open的才是最优解

THEN replace the old s in OPEN with the new lower f-value s

// we found a lower cost path to s, and we had not expanded s yet

// to guarantee the lowest cost solution path, we need to replace the old s in OPEN with the new lower-
// cost s

ELSE ignore s

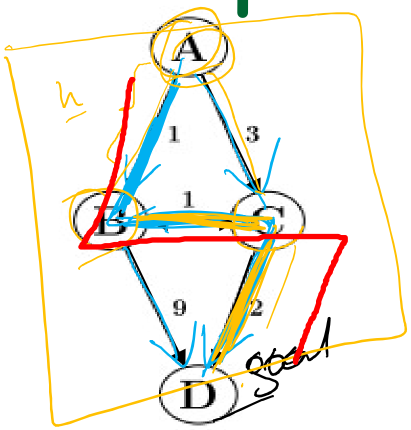
3. ELSE insert s in OPEN

// as usual

Example

ideal $h(n)$ $h_1(n)=h^*(n)$ $\forall n$

$\forall n \quad h(n) \le h^*(n)$



node	h_1	h_2	h^*
A	4	4	4
B	3	3	3
C	2	0	2
D	0	0	0

Solution paths
1. A B D -> cost of 10
2. A C D -> cost of 5
3. A B C D -> cost of 4
4. A C B D -> cost of 13

预计的小于等于实际的lowest cost to goal，admissible

- Admissibility -- h*(A)=4 h*(B)=3 h*(C)=2 h*(D)=0

  这些都看图就能统计出来

  - is h_1 admissible? Yes
  - is h_2 admissible? Yes

- Monotonic
  - is h_1 monotonic? Yes
    - $h_1(A) - h_1(B) \le g(B) - g(A)$ $\quad 4-3 \le 1-0 \quad 1 \le 1$
    - $h_1(A) - h_1(C) \le g(C) - g(A)$ $\quad 4-2 \le 2-0 \quad 2 \le 2$
    - $h_1(A) - h_1(D) \le g(D) - g(A)$ $\quad 4-0 \le 4-0 \quad 4 \le 4$
    - …

    每一pair的NODE与successor都要check，而不只是路径上的，但是g要取最小值

  - is h_2 monotonic? No

    不满足，不是Monotonic

    $h_2(A) - h_2(C) \nleq g(C) - g(A)$ $\quad 4-0 \nleq 2-0 \quad 3 \nleq 2$

    $h_2(B) - h_2(C) \nleq g(C) - g(B)$ $\quad 3-0 \nleq 2-1 \quad 3 \nleq 1$

    $\forall n \ s$

    最短路径上任意NS都要满足

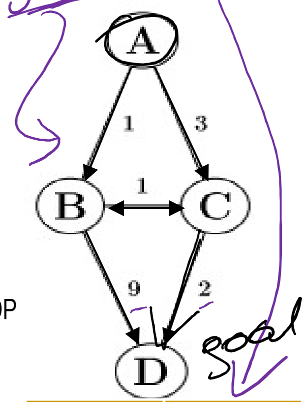
Example - h_2 admissible + not monotonic

node n

parent of n

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

	OPEN (unsorted… work in progress)	OPEN	CLOSED
1	• $A_{null}4_{g=0+h=4}$ 初始状态，f排序	1. $A_{null}4_{g=0+h=4}$	
2	• $B_A4_{g=1+h=3}$ • $C_A3_{g=3+h=0}$ sort 转成这个，SORT一边C在前面	1. $C_A3_{g=3+h=0}$ // we will explore C directly from A, but there is a lower cost path to C (ABC). h1 found it because it is monotonic, but h2 is not monotonic, so it could not guarantee that when we expand a node, we have found the lowest cost path to it… 2. $B_A4_{g=1+h=3}$	$A_{null}4_{g=0+h=4}$ ANUL POP
3	• $B_C7_{g=3+1+h=3}$ // B is already in OPEN (see below) but with a lower cost path. We do not replace the old, and ignore this version • $D_C5_{g=3+2+h=0}$ • $B_A4_{g=1+h=3}$ PUSHBD在前面后面无所谓，因为会SORT 左边是UNSORTED	1. $B_A4_{g=1+h=3}$ 2. $D_C5_{g=3+2+h=0}$ 右边是SORTED，这一步会处理OPEN多余	$A_{null}4_{g=0+h=4}$ $C_A3_{g=3+h=0}$ Cclose CLOSED是上一步得到的
4	• $C_B2_{g=1+1+h=0}$ // C was already in CLOSED but with a higher f-value, we just found a lower cost path to C… we need to put this version back in OPEN ;-(• $D_B10_{g=1+9+h=0}$ // D is already in OPEN (see below) but with a lower cost path. We do not replace the old, and ignore this version • $D_C5_{g=2+2+h=0}$	1. $C_B2_{g=1+1+h=0}$ 2. $D_C5_{g=3+2+h=0}$ C是closed但是被比下去了 排序选出优先D	$A_{null}4_{g=0+h=4}$ ~~$C_A3_{g=3+h=0}$~~ $B_A4_{g=1+h=3}$ C被删去
5	• $B_C6_{g=1+1+1+h=3}$ • $D_C4_{g=1+1+2+h=0}$ // D is already in OPEN (see below) but with a higher cost path. We replace the old version with version • $D_C5_{g=1+9+h=0}$ 未处理的D是上一个POP出去的successor加上 上一个SORTED OPEN	1. $D_C4_{g=1+1+2+h=0}$ 2. $B_C6_{g=1+1+1+h=3}$	$A_{null}4_{g=0+h=4}$ $B_A4_{g=1+h=3}$ $C_B2_{g=1+1+h=0}$
		goal(D_C4) = true! Solution path = $D_C C_B B_A A_{null}$ cost = 2+1+1 = 4 // lowest cost path found in 5 steps	



node	h_2
A	4
B	3
C	0
D	0

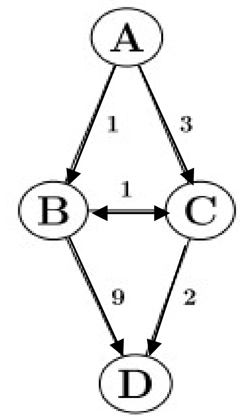
lowest cost path found in 5 steps

因为D的h等于0，D就是goal

Example – h_1 admissible + monotonic

	OPEN (unsorted… work in progress)	OPEN	CLOSED
1	• $A_{null}4_{g=0+h=4}$	1. $A_{null}4_{g=0+h=4}$	
2	• $B_A4_{g=1+h=3}$ • $C_A5_{g=3+h=2}$	1. $B_A4_{g=1+h=3}$ 2. $C_A5_{g=3+h=2}$	$A_{null}4_{g=0+h=4}$
3	• $C_B4_{g=1+1+h=2}$ // *C already in OPEN with a higher f-value, replace old version with this one* • $D_B10_{g=1+9+h=0}$ 新加入的CB4替代了老的CA5 • $C_A5_{g=3+h=2}$	1. $C_B4_{g=1+1+h=2}$ 2. $D_B10_{g=1+9+h=0}$	$A_{null}4_{g=0+h=4}$ $B_A4_{g=1+h=3}$
4	• $D_C4_{g=2+2+h=0}$ // *D already in OPEN with a higher f-value, replace old version with this one* • $B_C6_{g=2+1+h=3}$ // *B already in CLOSED but since h1 is monotonic, we do not need to check the f-value of the version in CLOSED because we know that the version in CLOSED will have a lower f-value, so can ignore this version* • $D_B10_{g=1+9+h=0}$	1. $D_C4_{g=2+2+h=0}$ 2. $B_C6_{g=2+1+h=3}$	$A_{null}4_{g=0+h=4}$ $B_A4_{g=1+h=3}$ $C_B4_{g=1+1+h=2}$
		goal(D_C4) = true! Solution path = $D_C\,C_B\,B_A\,A_{null}$ cost = 2+1+1 = 4 // *lowest cost path found in 4 steps*	

B已经在closed里了，但是是moto，没必要check，必然小于close



node	h_1
A	4
B	3
C	2
D	0

Admissible + Monotonic

lowest cost path found in 4 steps

LESS BACKTRACKING,只要四步

Informedness

rank the nodes

most promising ⟶ least promising

- Intuition:
  - $h(n) = 0$ for all nodes is less informed
  - number of misplaced tiles is less informed than Manhattan distance
- Formally:
  - given 2 admissible heuristics h_1 and h_2 // *ie.* $h_1(n) \leq h^*(n)$ *and* $h_2(n) \leq h^*(n)$ $\forall n$
    - if $h_1(n) \leq h_2(n)$, for all states n 越接近实际H*(n就是越Informed,就是越好
    - then h_2 is more informed than h_1
    - aka h_2 dominates h_1
- So?
  - a more informed heuristic expands fewer nodes 代表展开更少的点
  - aka the search path is shorter search path更少
  - however, you need to consider the computational cost of evaluating the heuristic… $h(n)$ 但是复杂度高的Hn用时可能也很高
  - the time spent computing heuristics must be recovered by a better search h2n dominate h1不代表更省时间

Today

1. State Space Representation
2. State Space Search
    a) Overview
    b) Uninformed search
        1. Breadth-first and Depth-first
        2. Depth-limited Search
        3. Iterative Deepening
        4. Uniform Cost
    c) Informed search
        1. Intro to Heuristics
        2. Hill climbing
        3. Greedy Best-First Search
        4. Algorithms A & A*
        5. More on Heuristics
    d) Summary

Summary

Search	Uses h(n)?	Uses g(n)?	OPEN list
Breadth-first	No	No	Priority queue sorted by level
Depth-first	No	No	Stack
Depth-limited	No	No	Stack
Iterative Deepening	No	No	Stack
Uniform Cost *- guarantees to find the lowest cost solution path*	No	Yes	Priority queue sorted by g(n) h(n) When generating successors: - If successor s already in OPEN with higher g(n), replace old version with new s - If successor s already in CLOSED, ignore s
Hill Climbing	Yes	No	N/A
Greedy Best-First *- no constraints on h(n)* *- no guarantee to find lowest cost solution path*	Yes	No	Priority queue sorted by h(n)
Algorithm A *- no constraints on h(n)* *- no guarantee to find lowest cost solution path*	Yes	Yes	Priority queue sorted by f(n) identical
Algorithm A* *- h(n) must be admissible* $\forall n\ h(n) \le h^*(n)$ *- guarantees to find the lowest cost solution path* provided	Yes	Yes	Priority queue sorted by f(n) If h(n) is NOT monotonic When generating successors: - If successor s already in OPEN with higher f(n), replace old version with new s - If successor s already in CLOSED with higher f(n), replace old version with new s If h(n) IS monotonic When generating successors: - If successor s already in OPEN with higher f(n), replace old version with new s - If successor s already in CLOSED, ignore it.

un informed

informed

Today

1. State Space Representation ✓
2. State Space Search ✓
   a) Overview ✓
   b) Uninformed search ✓
      1. Breadth-first and Depth-first ✓
      2. Depth-limited Search ✓
      3. Iterative Deepening ✓
      4. Uniform Cost ✓
   c) Informed search ✓
      1. Intro to Heuristics ✓
      2. Hill climbing ✓
      3. Greedy Best-First Search ✓
      4. Algorithms A & A* ✓
      5. More on Heuristics ✓
   d) Summary ✓

Up Next

1. Part 4: Adversarial Search