Лекция 3

Поиск в пространстве состояний:

Стратегии информированного (эвристического) поиска

Поиск «сначала лучший» (Best-First Search – *BFS*)

- Методы «слепого» поиска в большинстве случаев неэффективны, т. к.
  - …
- Эффективность поиска может быть повышена за счет *использования дополнительных специфичных для данного класса задач знаний* – ***эвристик***
- Эти знания:
  - должны позволять оценивать ***желательность раскрытия*** той или иной вершины в дереве поиска;
  - их естественно представить ***оценочной функцией***, которая возвращает число, отражающее предпочтительность раскрытия вершин;
- ***BFS*** (Best-First Search) – поиск, при котором *первой раскрывается вершина с максимальной оценкой*

Реализация BFS через обобщенный поиск

- В обобщенном алгоритме поиска единственным местом, где можно использовать дополнительные знания об особенностях задачи является функция построения очереди: **Queueing-Fn(Queue, Elements)**

- Пусть: ***Eval-Fn*** – функция оценки; ***Queuing-Fn*** – функция, упорядочивающая вершины в соответствии с ***Eval-Fn***

  Тогда BFS поиск на базе обобщенного поиска General-Seach реализуется следующим образом:

function BFS (*problem, Eval-Fn*) **returns** a solution sequence

  return GENERAL-SEARCH (*problem, Queuing-Fn)*

- Название BFS – строго говоря неточно, т. к. оценочная функция *не гарантирует* абсолютно лучшего, *оптимального выбора вершины* для раскрытия, а лишь определяет вершину, которая ***представляется лучшей*** *с точки зрения скорейшего достижения целевого состояния*

Жадный поиск (Greedy Search)

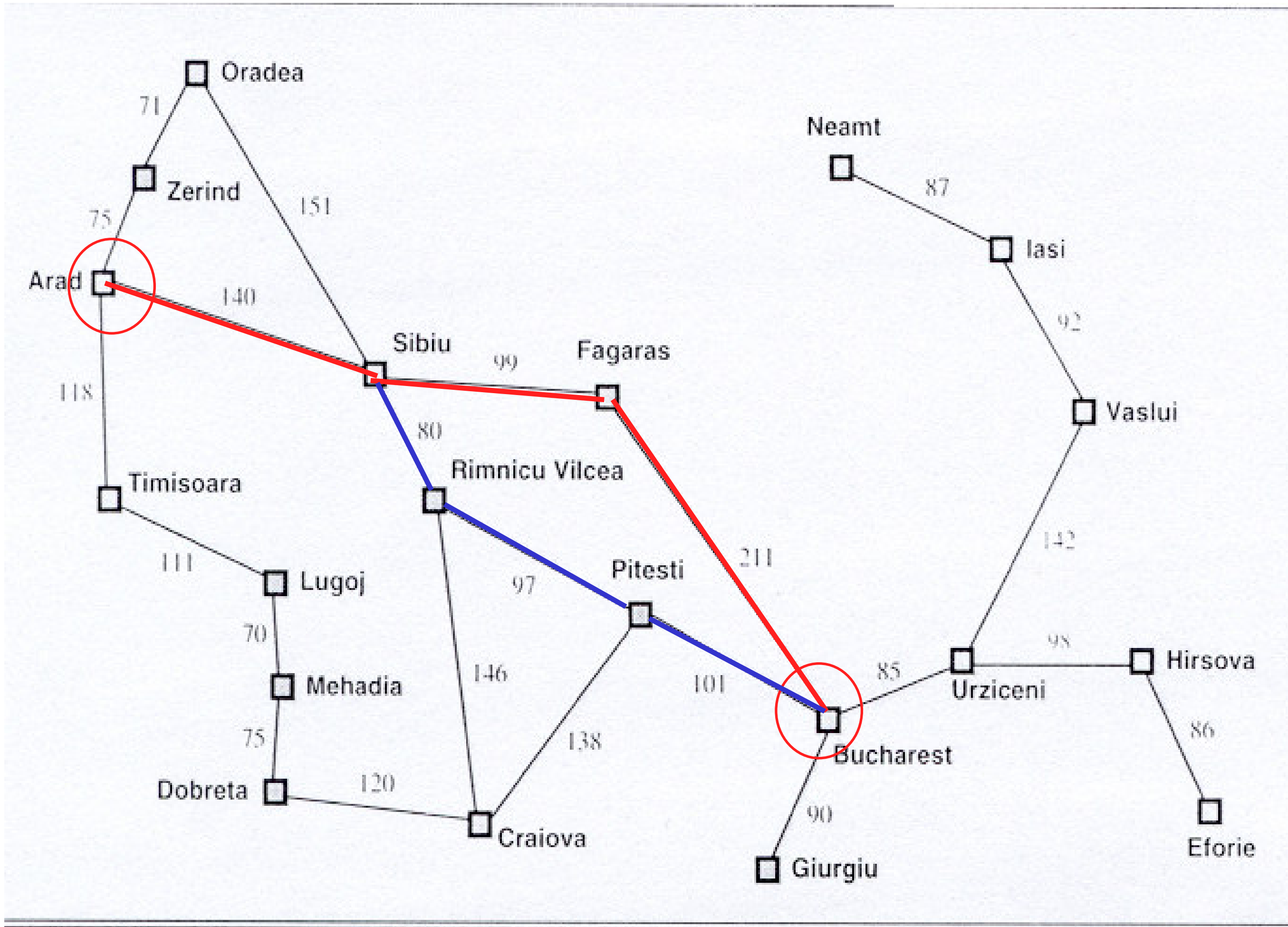
- **Жадный поиск** – первой раскрывается вершина, состояние которой *оценивается как ближайшее* к целевому состоянию
- Обозначим ***h*(*n*)** – ***оценочная*** *стоимость самого дешевого пути из состояния вершины n в целевое состояние*
- *h*(*n*) – эвристическая функция
- Тогда реализация жадного поиска на основе BFS :

function GREEDY-SEARCH (*problem*) **returns** a solution or failure

return BFS (*problem, h*)

Жадный поиск. Пример

В транспортных задачах (поиск путей на графе) в качестве *h*(*n*) часто используют расстояние по прямой - h_{SLD}

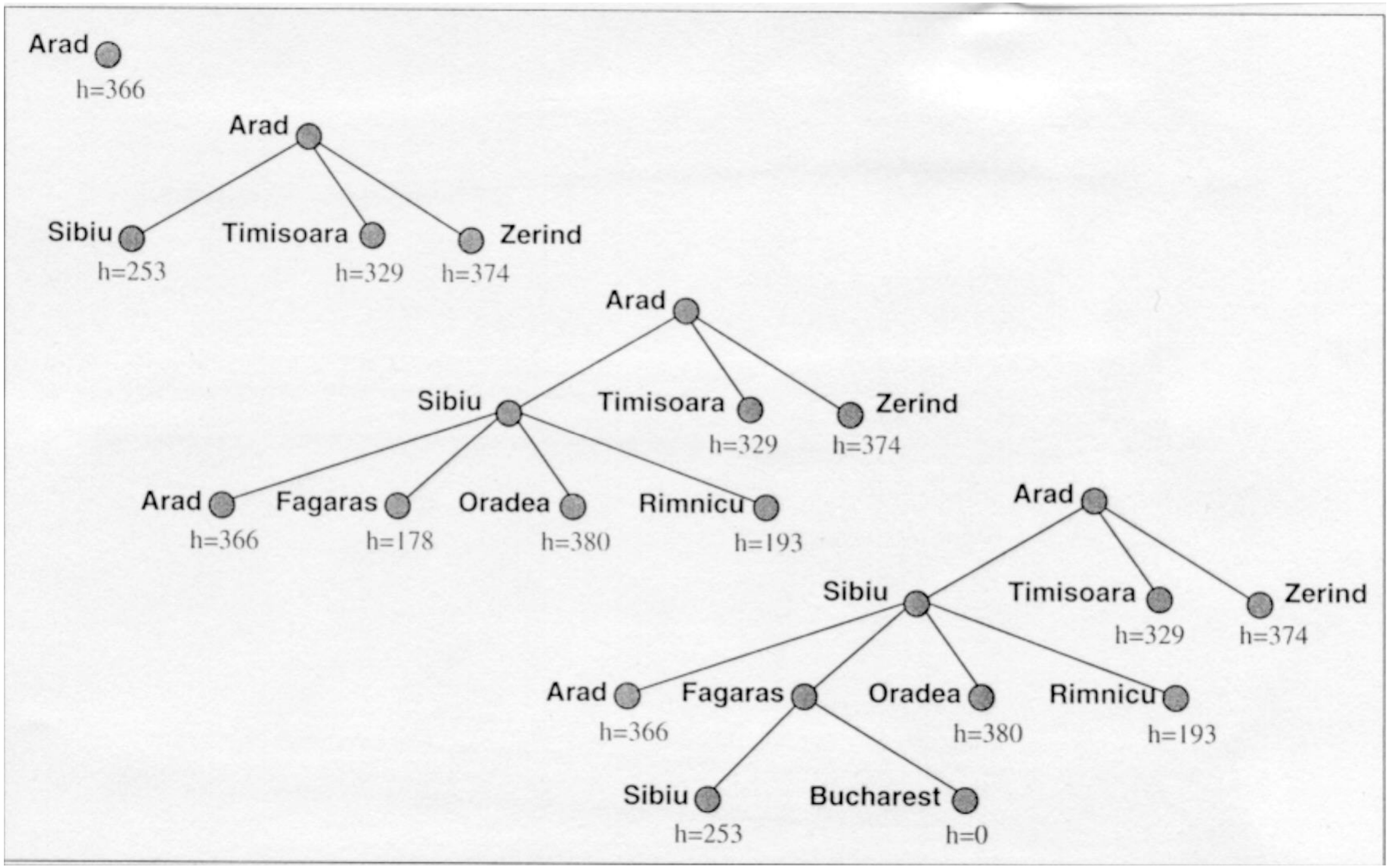


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Жадный поиск: пример

Шаги жадного поиска для Бухареста. В качестве эвристической функции используется h_{SLD} – расстояние по прямой до Бухареста



Эффективность жадного поиска

Жадный поиск:

- НЕ оптимален;
- НЕ полон;
- Временная сложность ***O***(b^m), m – максимальная глубина пространства поиска;
- Емкостная сложность ***O***(b^m), (все вершины сохраняются в памяти);

- аналогичен поиску в глубину;
- как правило находит решение (если оно существует) быстро, хотя оно не всегда является оптимальным;
- в конкретных задачах *при наличии хорошей эвристической функции* емкость и/или время могут быть существенно сокращены

Поиск A*

- *Жадный поиск* стремится *минимизировать оценочную стоимость пути до цели* $h(n)$, что позволяет в ряде случаев повысить эффективность поиска, однако не является ни оптимальным, ни полным

- С другой стороны *поиск по критерию стоимости* использует *минимальную стоимость пути до текущего состояния* $g(n)$ и является полным и оптимальным, однако часто оказывается неэффективным

- Естественно *совместить эти два подхода* или стратегии, чтобы использовать их преимущества

- Введем аддитивную оценочную стоимость:

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

$f(n)$ – *оценочная стоимость наиболее дешевого пути, проходящего через* n

Поиск A*

- Эвристика $h(n)$ является ***допустимой*** (admissible), если она *никогда не переоценивает реальное значение* (в задачах минимизации)
  - в задачах максимизации эвристика $h(n)$ допустима, если она *никогда не недооценивает реальное значение*

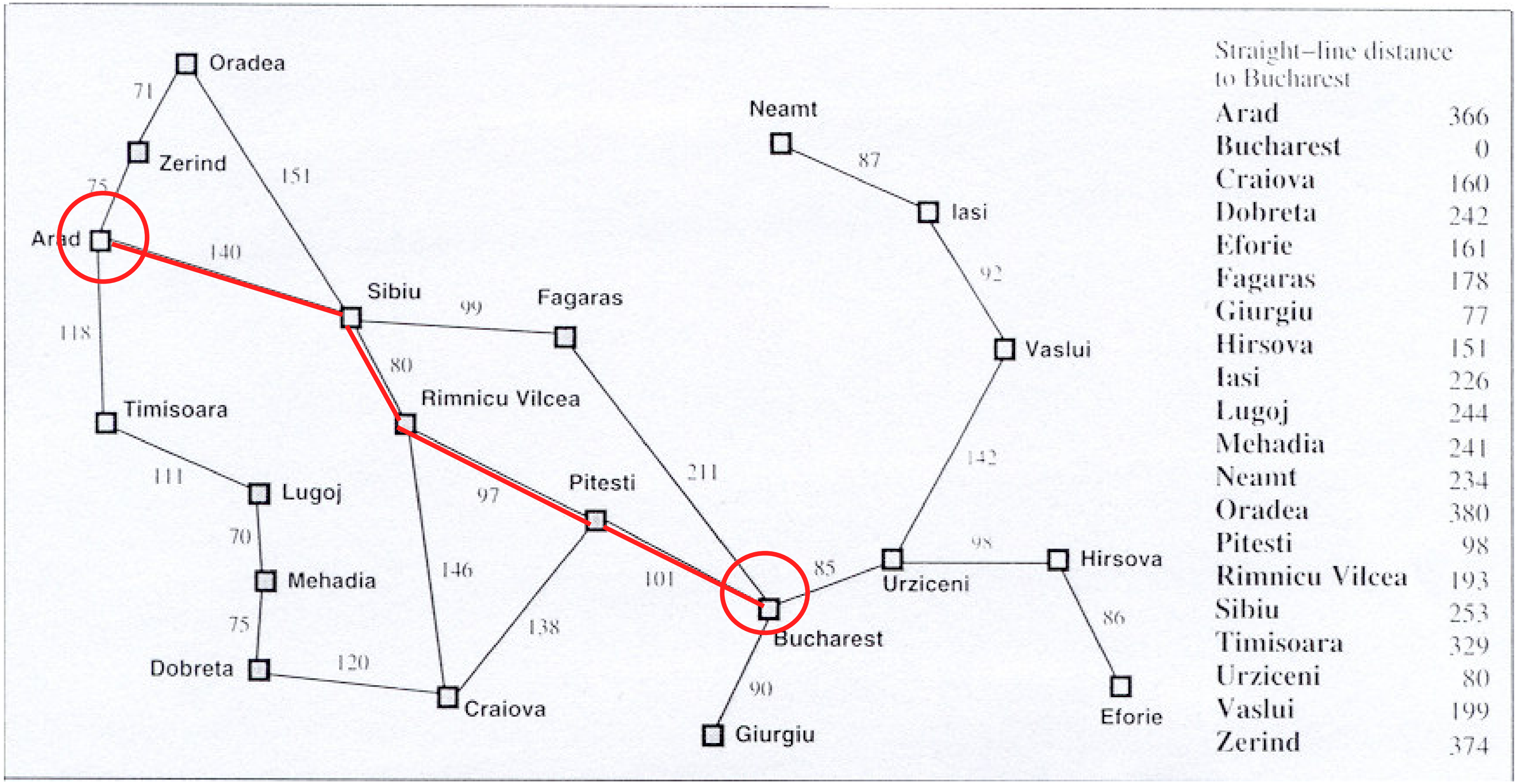
- Поиск BFS, использующий функцию $\boldsymbol{f(n) = g(n) + h(n)}$, где $h(n)$ является ***допустимой*** называется ***поиском A****

- Реализация поиска A* на основе BFS :

function A*-SEARCH (*problem*) **returns** a solution or failure
  return BFS (*problem, g+h)*

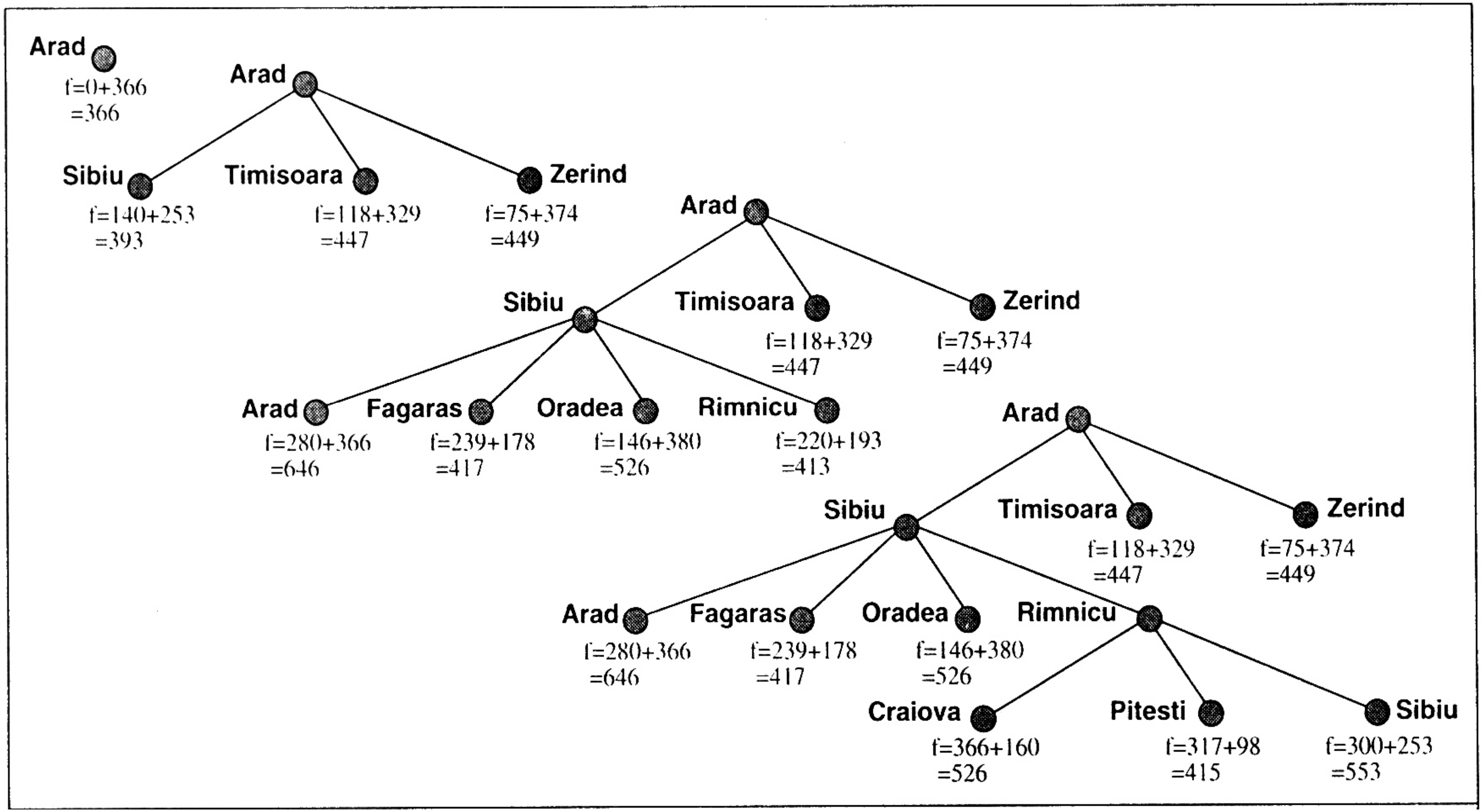
Поиск A*. Пример

Поиск A* в задаче поиска пути до Бухареста:



Поиск A*. Пример

Шаги поиска A* для пути в Бухарест. Вершины помечены *f = g + h*. *Значения h – расстояние по прямой до Бухареста*.



Монотонность эвристик

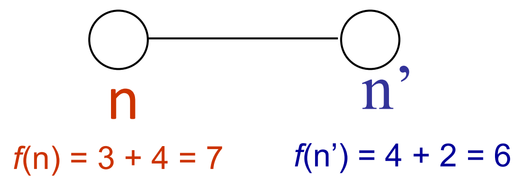
- $f(n)$ ***монотонна***, если для каждой вершины ***n*** и каждого ее последователя ***n'***, *сгенерированного любым действием* ***a***, оценочная стоимость достижения цели из ***n*** не больше чем стоимость $c(n, a, n')$ шага достижения ***n'*** плюс оценочная стоимость достижения цели из ***n'*** :

$$f(n) \le c(n, a, n') + f(n')$$

- Иначе говоря, монотонная $f(n)$ *никогда не убывает вдоль пути из корня* к цели

- Пример. Пусть $g(n) = 3$, $h(n) = 4$, т. е. $f(n) = 7$

  $g(n') = 4$, $h(n') = 2$, т. е. $f(n') = 6$



Имеем *нарушение монотонности*.

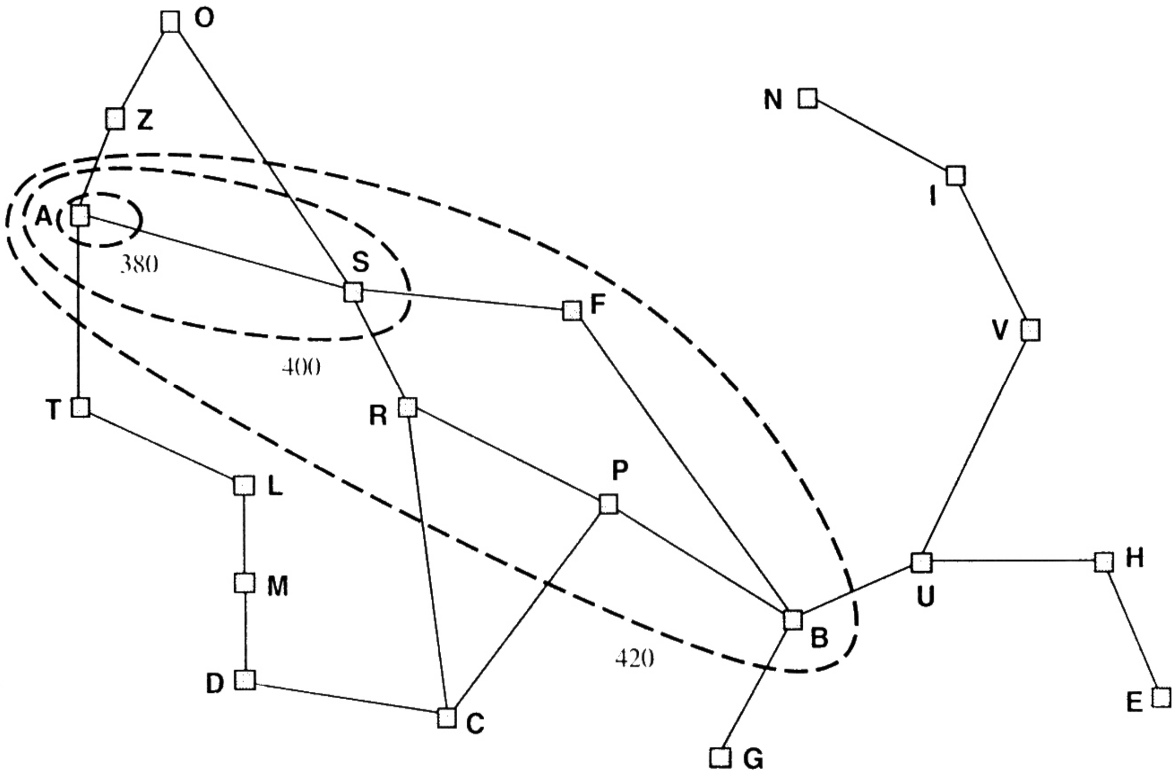
Для восстановления используется прием – выравнивание максимального пути:

$f(n') = \max(f(n), g(n') + h(n'))$

- Если эвристика ***монотонна***, то она ***является допустимой!!!***

Представление поиска A* контурами

A* поиск с монотонной эвристикой можно интерпретировать как поиск по контурам, которые *фокусируются в направлении целевой вершины*:



Контуры для *f* = 380, *f* = 400 и *f* = 420, для исходной точки Arad.
Вершины внутри контуров имеют *f*-стоимость меньше чем значение контура

Поиск A*

A^* поиск фокусируется в направлении целевой вершины.

Пусть f^* - стоимость оптимального пути, тогда поиск A^*:

- раскрывает все вершины, у которых $f(n) < f^*$
- может раскрывать некоторые вершины, у которых $f(n) = f^*$

A^* поиск является ***полным***, так как по мере расширения контура с возрастанием *f*, неизбежно достигается контур, у которого *f* равна стоимости f^* пути к целевому состоянию.

A^*-поиск является ***оптимально эффективным*** – никакой другой алгоритм *не гарантирует нахождения оптимальных вершин эффективнее, чем A^* поиск*

Полнота поиска A*

- Поскольку A* раскрывает вершины в порядке возрастания *f*, он рано или поздно должен достичь целевого состояния.

- Это справедливо, если число узлов с $f(n) \leq f^*$ *не бесконечно*
  - число узлов с $f(n) \leq f^*$ может быть бесконечно, если:
    - существует узел с бесконечным коэффициентом ветвления
    - существует путь с конечной стоимостью пути, но бесконечным числом узлов в нем

- Итак, A* является *полным* на графах с конечным коэффициентом ветвления при наличии некоторой положительной константы δ, такой что стоимость каждого оператора не меньше δ

Доказательство оптимальности поиска A*

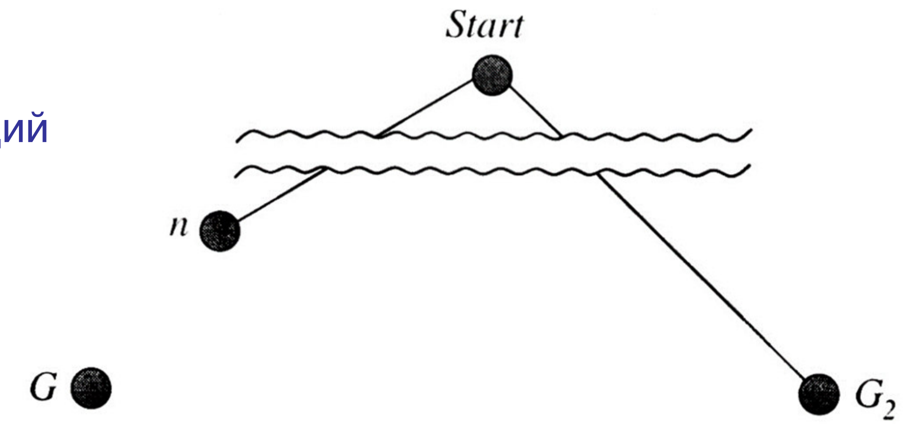
Пусть G – оптимальное целевое состояние и $f(G) = f^* = g(G)$.

Пусть G_2 – субоптимальное целевое состояние, т.е. $f(G_2) = g(G_2) > f^*$.

Предположим противное: A* выбрал из очереди G_2 (тогда A* завершится с субоптимальным решением)

Рассмотрим ситуацию, когда алгоритм мог бы (гипотетически!) достичь целевое состояние G_2 (субоптимальное) раньше, чем G (оптимальное):

n – вершина, которая в текущий момент является листом на оптимальном пути к G



Если *n* не выбрана для раскрытия раньше G_2, то $f(n) \geq f(G_2)$

Поскольку *h* является ***допустимой***, $f^* = f(G) \geq f(n)$.

Таким образом, $f^* \geq f(G_2)$. Поскольку $h(G_2)=0$, имеем $f^* \geq g(G_2)$ – противоречие!

Сложность A*

В общем случае:

- Временная - $O(b^d)$
- Емкостная - $O(b^d)$

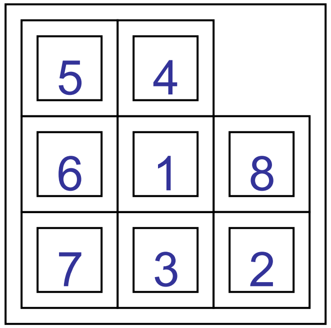
- Суб-экспоненциальный рост при $|h(n) - h^*(n)| \leq O(\log h^*(n))$
  - для большинства практических эвристик ошибка, к сожалению, по крайней мере пропорциональна стоимости пути

A* является **оптимально эффективным** для любой заданной *h*-функции среди алгоритмов расширяющих пути поиска от корня. Т.е никакой другой оптимальный алгоритм не гарантирует раскрытия меньшего числа вершин.

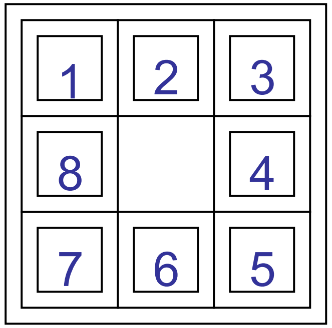
- Интуитивно ясно: любой алгоритм, который не раскрывает все вершины в контурах между корнем и контуром цели рискует пропустить оптимальное решение

Эвристики $h(n)$ для A*

Пример головоломки 8-ка



Исходное состояние



Целевое состояние

Эвристики:

- h_1 - число фишек, находящихся в неверной позиции
- h_2 сумма Манхэттенских расстояний фишек от их целевых позиций

- h_2 доминирует над h_1: $\forall n,\ h_2(n) \geq h_1(n)$

Эвристики *h*(*n*) для A*

Сравнение стоимости поиска и эффективного коэффициента ветвления для *поиска с итеративным углублением* и поиска A* с *h*1 и *h*2. Данные усреднены по 100 примерам 8-ки, для решений различной глубины

	Search Cost			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Эффективный коэффициент ветвления (*effective branching factor*) – среднее число преемников узла в дереве поиска после применения эвристик. Характеризует качество используемой эвристической функции.

Всегда лучше использовать эвристику *h*(*n*) с б**о**льшими значениями, при условии что она не делает переоценки, т.к. A* раскрывает все вершины с *f*(*n*) < *f* *