Система продукций для нахождения пути в графе

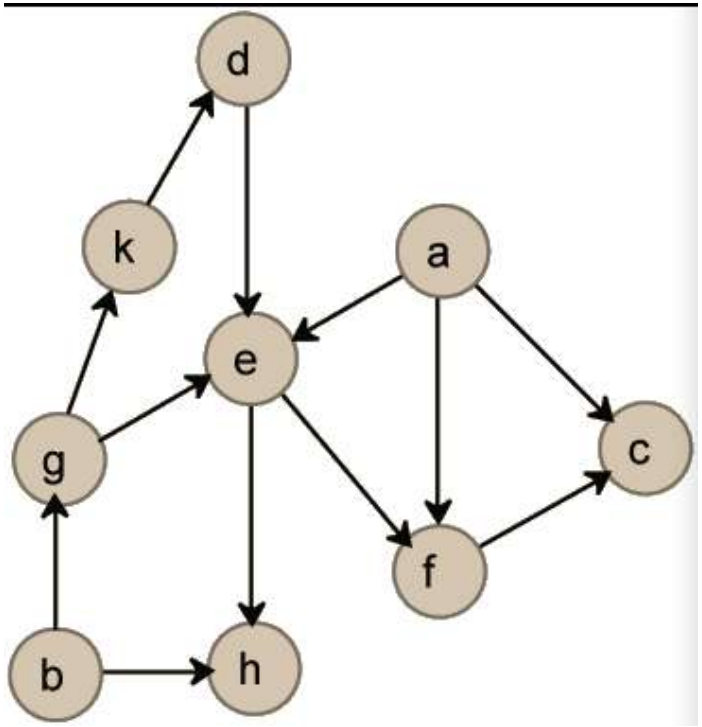
подготовила Слаповская Ирина, 451 группа

Формулировка задачи

- **Граф задан парами (a b) (из a можно попасть в b). Найти последовательность перемещений из начальной вершины в заданную.**

- Состояния пространства поиска – вершины графа.
- Правила – дуги графа, т.е. ориентированные ребра между вершинами.
- Множество исходных вершин – начальная вершина.
- Множество целевых вершин – конечная вершина.

Дуги *edges*




```
(setq *edges*
  '((A C)
    (F C)
    (A F)
    (E F)
    (A E)
    (D E)
    (E H)
    (K D)
    (G K)
    (B G)
    (B H)
    (G E)))
```

- Для работы с графом создаем переменную *edges* и записываем в неё список дуг заданного графа

Основная функция

```
(defun traverse (start goal &optional best)
  (if (and (atom start) (atom goal))
    (if best
      (pathbest (cons start '()) goal)
      (paths (cons start '()) goal))))
```

traverse – функция перемещения в графе.

Параметры, задаваемые пользователем:

- start – начальная вершина,
- goal – целевая вершина,
- best – необязательный аргумент, определяющий алгоритм поиска пути (поиск всех возможных путей / поиск наилучшего пути).

Если пользователь передает функции traverse два аргумента (исходная и целевая вершина), то осуществляется вызов функции paths, если присутствует третий аргумент (метка для запуска вычисления «жадного» алгоритма), то вызывается функция pathbest.

Функция paths

- Сперва рассмотрим случай, когда пользователем было передано два аргумента, и функция traverse осуществляет вызов рекурсивной функции paths. Результатом работы этой функции будет являться список всех возможных путей из начальной вершины в целевую.

Функция paths

```
(defun paths (open goal &optional beenlist)
  (cond
    ((null open) nil)
    ((member goal (children (car open) *edges*))
     (cons
       (reverse (cons goal (cons (car open) beenlist)))
       (append (paths (children (car open) *edges*) goal (cons (car open) beenlist))
               (paths (cdr (children (car open) *edges*)) goal (cons (car open) beenlist)))))
    (t (append (paths (children (car open) *edges*) goal (cons (car open) beenlist))
               (paths (cdr (children (car open) *edges*)) goal (cons (car open) beenlist))))))
```

Параметры функции paths:

- open – список неисследованных вершин (первоначально передается список из одной начальной вершины),
- goal – целевая вершина,
- beenlist – аргумент, хранящий путь до текущей вершины (первоначально nil, затем в ходе работы функции наполняется значениями).

Функция paths

```
(defun paths (open goal &optional beenlist)
  (cond
    ((null open) nil)
    ((member goal (children (car open) *edges*))
     (cons
       (reverse (cons goal (cons (car open) beenlist)))
       (append (paths (children (car open) *edges*) goal (cons (car open) beenlist))
               (paths (cdr (children (car open) *edges*)) goal (cons (car open) beenlist)))))
    (t (append (paths (children (car open) *edges*) goal (cons (car open) beenlist))
               (paths (cdr (children (car open) *edges*)) goal (cons (car open) beenlist))))))
```

Алгоритм работы функции:

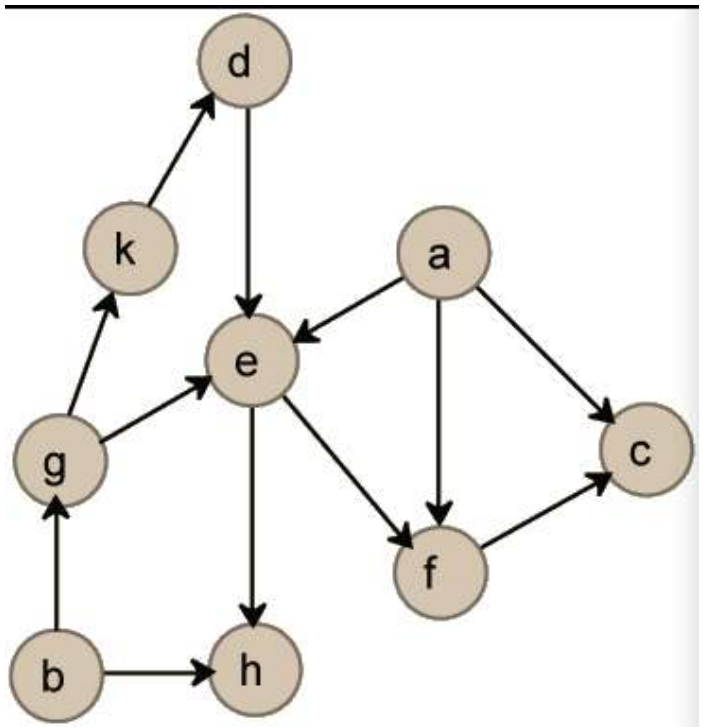
- Если список open пуст, то работа функции прерывается (результат - nil).
- Если целевая вершина является потомком первой вершины в списке open, то формируется путь до цели. Затем рекурсивно вызываются две функции paths для рассмотрения потомков данной вершины.
- При невыполнении ни одного из условий происходит вызов описанной выше рекурсии.

Функция children

```
(defun children (root edges)
  (cond
    ((null edges) nil)
    ((eq root (caar edges))
     (cons (cadar edges) (children root (cdr edges))))
    (t (children root (cdr edges)))))
```

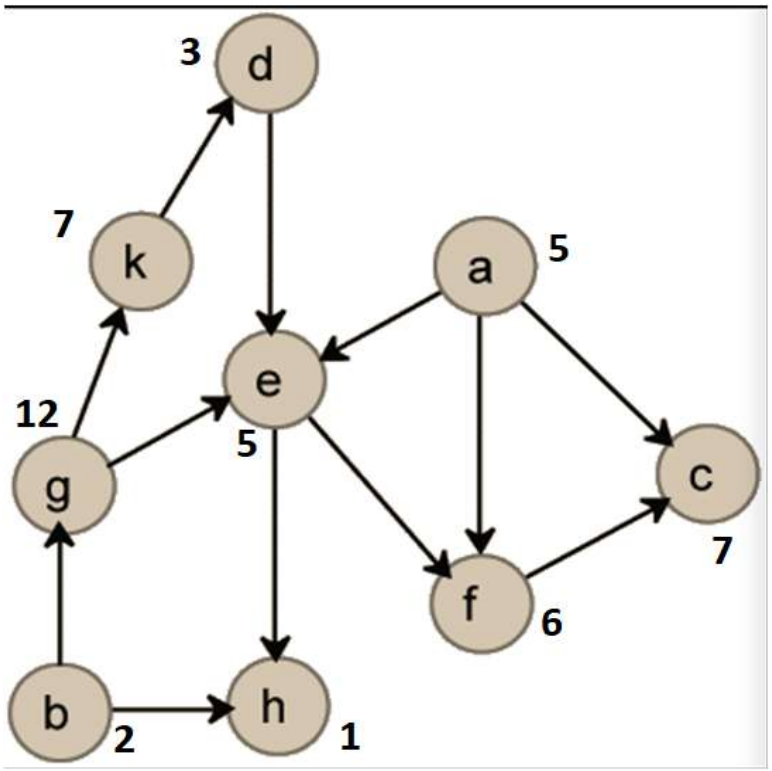
- В ходе работы функции paths используются результаты функции children, которая генерирует список потомков вершины на основе списка смежности графа.

Работа функции paths как результат вызова traverse для двух параметров




```
TRAVERSE

CL-USER 5 > (traverse 'B 'C)
((B G K D E F C) (B G E F C))

CL-USER 6 > (traverse 'B 'H)
((B H) (B G K D E H) (B G E H))

CL-USER 7 > (traverse 'C 'A)
NIL

CL-USER 8 > (traverse 'D 'F)
((D E F))
```

Функция pathbest. Оценки *grades*




```
(setq *grades*
  '((A 5)
    (B 2)
    (C 7)
    (D 3)
    (E 5)
    (F 6)
    (G 12)
    (H 1)
    (K 7)))
```

- Для поиска наилучшего варианта пути в данной системе продукций применяется функция pathbest. Для осуществления данного алгоритма необходимо сохранить эвристические оценки вершин в переменную *grades*. Будем считать вершину с большей оценкой наиболее предпочтительной.

Функция pathbest

```
(defun pathbest (open goal &optional beenlist ends)
  (cond
    ((eq (car open) goal) (reverse (cons goal beenlist)))
    ((null (notends (children (car open) *edges*) ends))
     (pathbest beenlist goal (cdr beenlist) (cons (car open) ends)))
    (t (pathbest
         (sortopen (notends (children (car open) *edges*) ends))
         goal
         (cons (car open) beenlist)
         ends))))
```

Алгоритм работы функции:

- Если первая вершина списка open является целевой, то формируется список до этой вершины, и происходит завершение работы функции.
- Если у первой вершины списка open нет потомков, не являющихся тупиковыми состояниями (ends), то происходит рекурсивный вызов функции pathbest для предыдущей вершины.
- В ином случае, осуществляется рекурсия для отсортированного списка потомков текущей вершины.

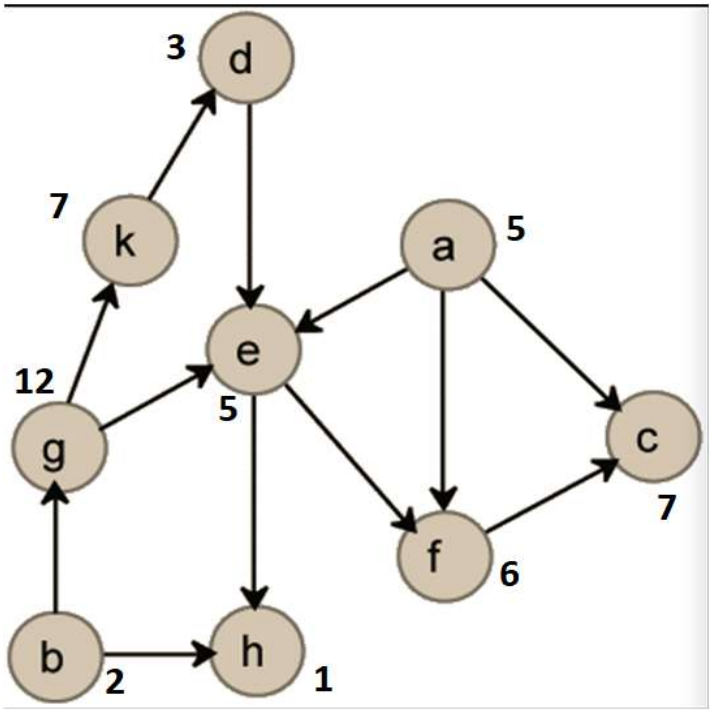
Функция pathbest

```
(defun pathbest (open goal &optional beenlist ends)
  (cond
    ((eq (car open) goal) (reverse (cons goal beenlist)))
    ((null (notends (children (car open) *edges*) ends))
     (pathbest beenlist goal (cdr beenlist) (cons (car open) ends)))
    (t (pathbest
         (sortopen (notends (children (car open) *edges*) ends))
         goal
         (cons (car open) beenlist)
         ends))))
```

Реализация функции pathbest включает вызовы следующих пользовательских функций:

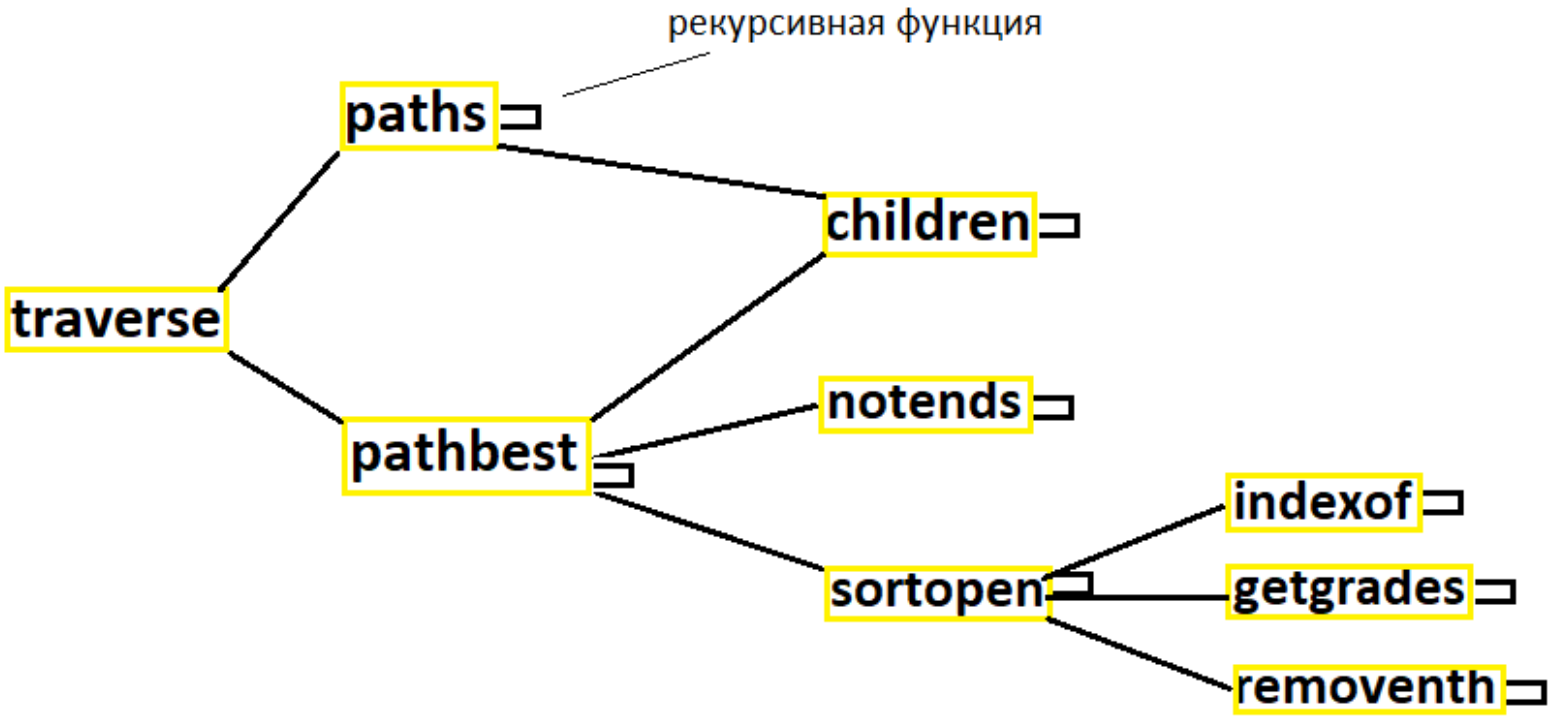
- sortopen – сортирует список по критерию наилучшей оценки,
- notends – убирает из списка потомков вершины, однозначно не ведущие в целевую,
- children

Работа функции pathbest




```
CL-USER 22 : 1 >  (traverse 'B 'C 'best)
(B G K D E F C)

CL-USER 23 : 1 >  (traverse 'B 'H 'best)
(B G K D E H)

CL-USER 24 : 1 >  (traverse 'A 'C 'best)
(A C)
```

Схема вызовов пользовательских функций в системе продукций



Функция notends

```
(defun notends (children ends)
  (cond
    ((null children) nil)
    ((member (car children) ends) (notends (cdr children) ends))
    (t (cons (car children) (notends (cdr children) ends)))))
```

- notends – убирает из списка потомков тупиковые вершины

Функция sortopen

```
(defun sortopen (open)
  (cond
    ((null open) nil)
    (t (cons
        (nth (indexof (apply 'max (getgrades open)) (getgrades open) 0) open)
        (sortopen (removenth (indexof (apply 'max (getgrades open)) (getgrades open) 0) open))))))
```

- sortopen – сортирует список по критерию максимальной оценки

Вспомогательные функции для sortopen. Функция getgrades

```
(defun getgrades (open)
  (cond
    ((null open) nil)
    (t (append (cdr (assoc (car open) *grades*)) (getgrades (cdr open))))))
```

- getgrades – список оценок для списка вершин

Вспомогательные функции для sortopen. Функция removenth

```
(defun removenth (n list)
  (cond
    ((or (zerop n) (null list)) (cdr list))
    (t (cons (car list) (removenth (- n 1) (cdr list))))))
```

- removenth – удаляет n-ый элемент из списка

Вспомогательные функции для sortopen. Функция indexof

```
(defun indexof (element list n)
  (cond
    ((null list) nil)
    ((eq element (car list)) (+ n 0))
    (t (indexof element (cdr list) (+ n 1)))))
```

- indexof – индекс элемента в списке

Спасибо за внимание