Курс «Алгоритмы и алгоритмические языки»
1 семестр 2014/2015

Лекция 16

Описание алгоритма топологической сортировки на языке Си

```
#include <stdio.h>
#include <stdlib.h>
typedef struct ldr {    /*дескриптор ведущего узла*/
  char key;
  int count;
  struct ldr *next;
  struct trl *trail;
} leader;
typedef struct trl {    /*дескриптор ведомого узла*/
  struct ldr *id;
  struct trl *next;
} trailer;

leader *head, *tail;    /*два вспомогательных узла*/
int lnum;               /*счетчик ведущих узлов*/
```

Описание алгоритма топологической сортировки на языке Си

```
/* поиск по ключу w */

leader *find (char w) {
  leader *h = head;
  /* "барьер" на случай отсутствия w */
  tail->key = w;
  while (h->key != w)
    h = h->next;
  if (h == tail) {
    /* генерация нового ведущего узла */
    tail = (leader *) malloc (sizeof (leader));
    /* старый tail становится новым элементом списка */
    lnum++;
    h->count = 0;
    h->trail = NULL;
    h->next = tail;
  }
  return h;
}
```

Описание алгоритма топологической сортировки на языке Си

```
void init_list() {
  /* инициализация списка «ведущих» */
  leader *p, *q;
  trailer *t;
  char x, y;

  head = (leader *) malloc (sizeof (leader));
  tail = head;
  lnum = 0;          /* начальная установка */
  while (1) {
    if (scanf ("%c %c", &x, &y) != 2)
      break;
    /* включение пары в список */
    p = find (x);
    q = find (y);
    /* коррекция списка */
    t = (trailer *) malloc (sizeof (trailer));
    t->id = q;
    t->next = p->trail;
    p->trail = t;
    q->count += 1;
  }
}
```

Описание алгоритма топологической сортировки на языке Си

```
/* Исходный список построен. Организация нового списка */
void sort_list() {
  leader *p, *q;
  trailer *t;
  /* В выходной список включаются все узлы старого
     с count == 0 */
  p = head;
  head = NULL; /* голова выходного списка */
  while (p != tail) {
    q = p;
    p = q->next;
    if (q->count == 0) {
      /* включение q в выходной список */
      q->next = head;
      head = q;
    }
  }
<...>
```

Описание алгоритма топологической сортировки на языке Си

```
/* Фаза сортировки и вывода результатов из нового списка */


<...>
  q = head; /* есть ведущий узел -> head != NULL */
  while (q != NULL) {
    printf ("%c\n", q->key);
    lnum--;
    t = q->trail;
    q = q->next;
    while (t != NULL) {
      p = t->id;
      p->count -= 1;
      if (p->count == 0) {
        p->next = q;
        q = p;
      }
      t = t->next;
      /* здесь можно удалить ведомый элемент */
    }
  }
  /* lnum == 0 */
}
```

Описание алгоритма топологической сортировки на языке Си

```
int main() {
  init_list ();
  sort_list ();
  return 0;
}
```

Сортировка

- ***Постановка задачи***

  Сортировка – это упорядочение наборов однотипных данных, для которых определено отношение линейного порядка (например, <) по возрастанию или по убыванию.
  Здесь будут рассматриваться целочисленные данные и отношение порядка "<".

- В стандартную библиотеку **`stdlib`** входит функция **`qsort`**:

  ```
  void qsort (void *buf, size_t num, size_t size,
  int(*compare)(const void *, const void *));
  ```

  Функция **`qsort`** сортирует (по возрастанию) массив с указателем **`buf`**, используя алгоритм быстрой сортировки *Ч.Э.Р. Хоара*, который считается одним из лучших алгоритмов сортировки общего назначения.

  Параметр **`num`** задает количество элементов массива **`buf`**, параметр **`size`** – размер (в байтах) элемента массива **`buf`**.

  Параметр **`int(*compare)(const void *, const void *)`**задает правило сравнения элементов массива **`num.`**

Сортировка

- Функция, указатель на которую передается в **`qsort`** в качестве аргумента, соответствующего параметру **`int(*compare)(const void *, const void *)`**, должна иметь описание:

  `int` ***`имя функции`*** **`(const void *`*`arg1`**`, const void *`*`arg2`**`)`**

  и возвращать:

  - целое < 0, если ***`arg1`*** < ***`arg2`***,
  - целое = 0, если ***`arg1`*** = ***`arg2`***
  - целое > 0, если ***`arg1`*** > ***`arg2`***

Сложность алгоритмов

- ***Размер входа*:** числовая величина, характеризующая количество входных данных (например – длина битовой записи чисел- параметров алгоритма)
- ***Сложность в наихудшем случае*:** функция размера входа, отражающая максимум затрат на выполнение алгоритма для данного размера
  - временнАя сложность
  - пространственная сложность (затраты памяти)
  - часто оценивают не все затраты, а только самые “дорогие” операции
- ***Сложность в среднем*:** функция размера входа, отражающая средние затраты на выполнение алгоритма для входа данного размера (учет вероятностей входа)
- ***Асимптотические оценки сложности*:** O-нотация (оценка сверху), точная O-оценка, Θ-оценка.
- Подробности: С.А. Абрамов. Лекции о сложности алгоритмов. М.: МЦНМО, 2009

Сортировка

- ***Простейший алгоритм сортировки***: сведение сортировки к задаче нахождения максимального (минимального) из *n* чисел.

  Нахождение максимума *n* чисел (*n* сравнений):

  Числа содержатся в массиве **`int a[n];`**

```
max = a[0];
for (i = 1; i < n; i++)
  if (a[i] > max)
    max = a[i];
```

- ***Алгоритм сортировки***: находим максимальное из *n* чисел, получаем последний элемент отсортированного массива (*n* сравнений);
  находим максимальное из *n* – 1 оставшихся чисел, получаем предпоследний элемент отсортированного массива (еще *n* – 1 сравнений); и так далее.
- ***Общее количество сравнений***: $1 + 2 + \ldots + n\text{-}1 + n = n(n - 1)/2$.

  Сложность алгоритма $O(n^2)$.