Графи

Изготвил:
гл.ас. д-р Нора Ангелова

Графи

- Множество от върхове заедно с множество от ребра.
- Всяко ребро се задава чрез двойка върхове
- Двойките могат да бъдат наредени – ориентиран граф, или ненаредени – неорентиран граф
- Ребрата могат да се свързват с етиките - тегла

Графи

Физическо представяне:

- Последователно
- Свързано

Графи
(последователно представяне)

- Матрица на съседство

n – брой върхове в графа

матрицата е с размерност nxn

Ако (i,j) е дъга, елементът в i-ти ред и j-ти стълб на матрицата е 1, в противен случай той е 0.

Графи
(последователно представяне)

Недостатъци:

- При големи графи матриците на съседство заемат прекалено много памет и при много алгоритми водят до квадратично време за изпълнение.
- Матрици на съседство, с голям брой нулеви елементи, се наричат **разредни**.

Графи
(свързано представяне)

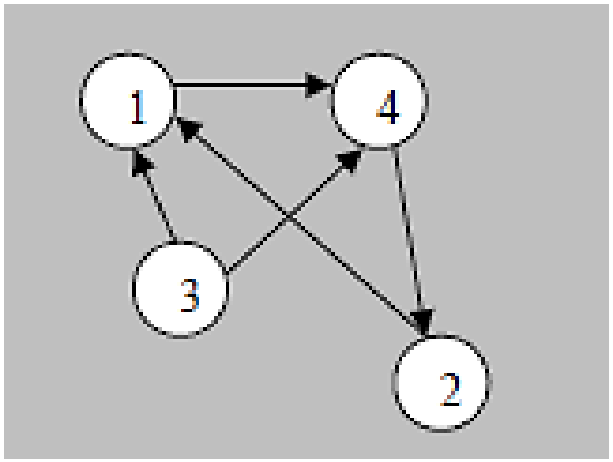
- Свързани списъци

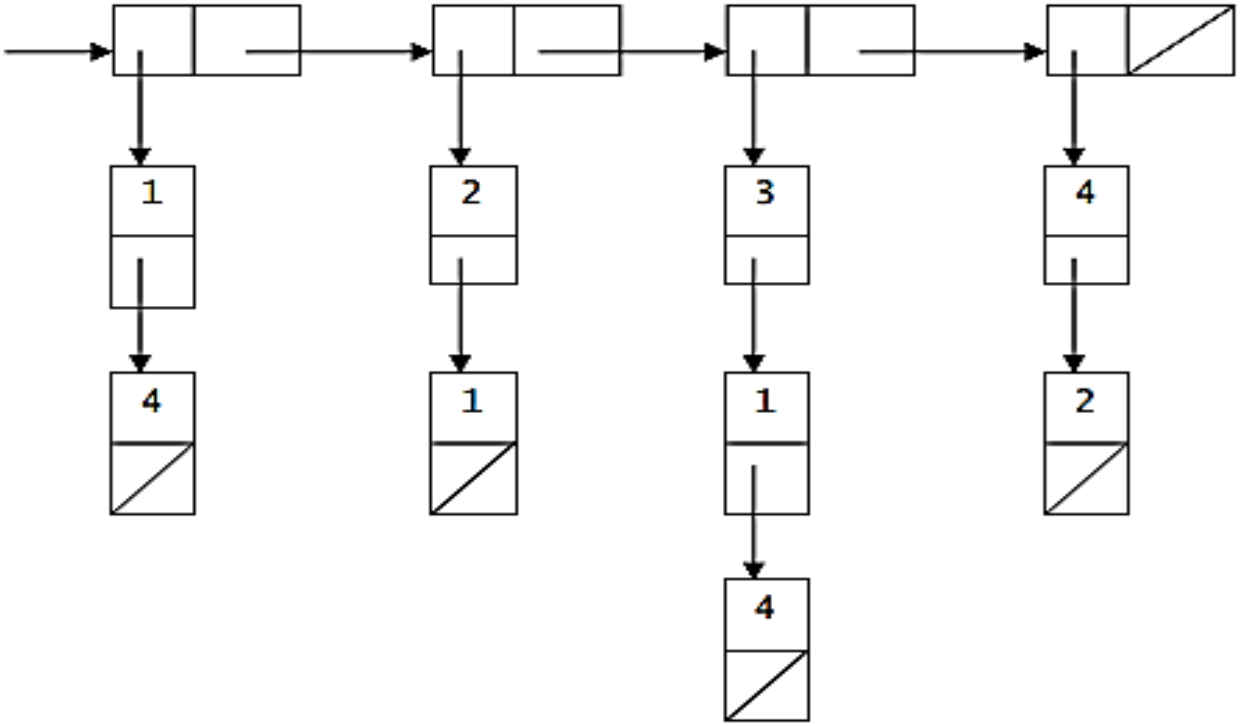
`n – брой върхове в графа`

`Свързан списък с n елемента`

- Елементите на списъка са списъци като всеки подсписък започва с връх `i` на графа и съдържа всички върхове `j`, така че има дъга от `i` до `j`.

- Ще използваме само ориентирани графи. Неориентираните графи ще представяме като ориентирани.


1
4
3
2


1
2
3
4
4
1
1
2
4

Графи
(свързано представяне)

- Това представяне се реализира чрез шаблона на класа `LList`, дефиниращ свързан списък с една връзка.

Специализациите

```
typedef LList<int> IntList;
typedef LList<IntList> IntGraph;
```

определят класа `IntGraph`, задаващ граф с цели числа.

Графи
(свързано представяне)

- Създаването на графа

Алгоритъм:
Докато желаем, въвеждаме цяло число, означаващо връх на графа, след което го включваме като връх. След това отново докато желаем въвеждаме наредени двойки от върхове, означаващи дъги в графа и ги включване.

Графи

(свързано представяне)

Създаването на графа

- Функция `Point` - намира указател към двойната кутия, в информационната част на която е записан върхът a на графа g.

```
elem_link<int>* Point(int a, IntGraph &g)
{
  g.IterStart();
  elem_link<IntList>*p;
  elem_link<int> *q;
  do
  {
    p = g.Iter();
    p->inf.IterStart();
    q = p->inf.Iter();
  } while(q->inf != a);
  return q;
}
```

Графи
(свързано представяне)

Създаването на графа

- Функция `Point -` намира указател към двойната кутия, в информационната част на която е записан върхът a на графа g.
  В графа g има връх a.

Графи

(свързано представяне)

```
void AddTop(int a, IntGraph &g)
{
  IntList l;
  l.ToEnd(a);
  g.ToEnd(l);
}

void AddRib(int a, int b, IntGraph &g)
{
  elem_link<int> * q = Point(a, g), *p;
  while (q->link) q = q->link;
  p = new elem_link<int>;
  p->inf = b;
  p->link = NULL;
  q->link = p;
}
```

Графи
(свързано представяне)

```
void create_graph(IntGraph &g)
{
  char c;
  do
  {
    cout << "top_of_graph: ";
    int x; cin >> x;
    AddTop(x, g);
    cout << "Top y/n? "; cin >> c;
  } while (c == 'y');

  cout << "Ribs:\n";

  do
  {
    cout << "start top: ";
    int x; cin >> x;
    cout << "end top: ";
    int y; cin >> y;
    AddRib(x, y, g);
    cout << "next: y/n? "; cin >> c;
  } while (c == 'y');
}
```

Графи
(свързано представяне)

Извеждане на графа

```
void LList<IntList>::print()
{
  elem<IntList> *p = Start;
  while (p)
  {
    p->inf.print();
    p = p->link;
  }
  cout << endl;
}
```

```
cout << “КРАЙ”;
```