суын кәсіби қызметте қолдануға жеткіліксіз дәрежеде бейімдейді, студенттер осы алған білімдерін қашан және қайда қолдануға болатынын білмейді.

Сондықтан да мультимедиалық технологияларды қолданылуының басты себебі қазіргі уақытта оқытушылар студенттер бойында сабақ барысында оқуға деген қызығушылық деңгейінің төмендеуімен, өз бетінше жұмыс істеудегі салғырттық, тіпті оқуға деген қызығушылықтары болмауымен байланысты мәселелерге тап болуда.

Көптеген зерттеулерде математиканы оқытудың кәсіби бағытын қалыптастыруға кәсіби бейімделген есептерді өңдеу және шешу, математиканы оқытуда жаңа белсенді әдістерді қолдану, ақпараттық және коммуникациялық технологиялардың құралдарын қолдану болып табылады.

МОЖ құрылымына математикалық түсініктерді абстракциялау үшін иллюстративті блоктар ену керек, ол арқылы оқытушы дәрісте визуалды ақпараттарды көрсете алады. Соның ішінде МОЖ математикадан дәрісте мына бөлімдерді оқытуда қолданылады: «Жиын», «Айналу фигуралары», «Үшінші ретті интегралдар», Фурье қатарлары», «Фурье интегралдары». Сонымен қатар математикадан МОЖ мына құрылымнан тұруы мүмкін: иллюстративті, ақпараттық-энциклопедиялық, түсіндірмелі, жаттығушы, тестілеу, бақылау.

ӘДЕБИЕТТЕР ТІЗІМІ

1. Нурбекова Ж.К., Даутова А.З., Кашкинбаева Д.Б. Технология проектирования мультимедийных обучающих систем: Учебно-методические пособие. – Павлодар, 2003 – 108 с.
2. Семенова, Н.Г. Структура мультимедийной обучающей системы по дисциплине «Высшая математика» / Н.Г. Семенова, И.П. Томина // Ученые записки. Вып. 31. – М.: ИИО РАО, 2009. – С. 153-157.

ТЕОРИЯ ГРАФОВ КАК ОДИН ИЗ МЕТОДОВ РАЗРАБОТКИ АЛГОРИТМОВ В ПРОГРАММИРОВАНИИ

GRAPH THEORY AS ONE OF METHODS DEVELOPING ALGORITHMS IN PROGRAMMING

Цыганова А.Д.
Костанайский государственный педагогический институт, г.Костанай, Казахстан

Решение задач повышенной сложности – необходимая предпосылка глубокого понимания основ предмета. В каждом классе найдётся обычно небольшая, но очень важная группа сильных учащихся. Сильные ученики способны работать самостоятельно, обладают высокой мотивацией к учению, любознательностью, упорством. Именно им, в первую очередь, нужны хорошие задачи с указаниями к решению или готовыми решениями.

Книг по олимпиадной информатике (алгоритмам, необходимым для решения олимпиадных задач), доступных для понимания учащимся 7-10 классов (а именно в это время большинство участников олимпиад начинают решать задачи) очень мало. Но и алгоритмы – не единственное, что необходимо знать при решении задачи – нужен также «стиль», «хитрые» приёмы и другие знания, которые сложно получить из «классических» книг по программированию. Всё это доступно объяснить ребёнку может лишь компетентный учитель, который сам в должной мере разбирается в этом вопросе. Изучение методов решения олимпиадных задач будущим педагогом поможет ему впоследствии понять, какие трудности с пониманием алгоритма решения поставленной задачи могут возникнуть у школьника, соответственно сконцентрировать внимание на сложных и спорных моментах.

Педагог, ещё на этапе своего обучения, изучая олимпиадные задачи, знакомясь с методами их решения, теориями и навыками, в будущем сможет объяснить своим ученикам принципы их действия, ответить на возникающие вопросы, научит их видеть в задаче.

Идеи и методы, используемые при решении задач олимпиадного характера, могут быть применены при решении широкого круга практических задач, в которых необходимо организовать эффективную обработку большого объёма информации в условиях ограниченности доступных ресурсов.

Анализ стартового уровня подготовленности первокурсников в области программирования позволяют сделать вывод о том, что подавляющая масса выпускников школ не имеет должного уровня подготовки по программированию из-за отсутствия профессиональной компетентности учителя информатики в области программирования и, уж тем более, по решению задач олимпиадного характера.

Учитывая это, для специальности педвуза «Информатика» введены в учебные планы такие курсы по выбору, как «Алгоритмы на графах», «Комбинаторика для программистов», «Нестандартные задачи в информатике», «Олимпиадная информатика», «Фрактальная графика». Учебного, а тем более, методического материала по данным курсам нет; используется разработанный и накопленный преподавателями кафедры материал и опыт при работе с «олимпийским резервом», анализ решения олимпиадных заданий различного уровня.

На кафедре была проведена работа по подготовке и изданию учебно-методических пособий в помощь студентам – будущим учителям информатики, в которых собран материал, необходимый для подготовки ребят к олимпиадам, начиная «с нуля», проведена классификация задач по основным разделам, рассмотрены основные алгоритмы и приёмы решения олимпиадных задач различного уровня – «Практикум решения задач по информатике», в котором обобщается многолетний опыт работы авторов в подготовке участников-призёров олимпиад различного уровня и учебное пособие по самому сложному и неизученному разделу программирования «Алгоритмы на графах», подготовлен материал для издания учебно-методического пособия по курсу «Олимпиадная информатика».

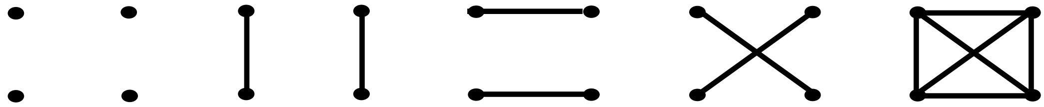
«Олимпиадная информатика» – это «увлечение» для тех, кто считает, что программирование – это, прежде всего, искусство решения логически сложных задач. Главная задача учителя – научить мыслительной деятельности в таком виде, в каком она используется на самом деле: с ошибками, тупиковыми вариантами, рождением красивой идеи. И что, пожалуй, ещё важнее – показать возможность такой организации своей мыслительной деятельности, при которой поиск решения становится деятельностью системной и планомерной.

Если взять интересную, творческую задачу, то может оказаться, что определить, к какому типу она принадлежит, довольно сложно. Часто задача может относиться и не к одному, а к нескольким типам. Зачастую мы имеем дело с нестандартными (нетипичными) задачами, не подчиняющимися какой-либо классификации, требующими творческого подхода, самостоятельного придумывания и разработки алгоритма.

Одним из методов оптимизации алгоритма решения задачи является использование графовых моделей алгоритмов решения задачи. Решение задач с использованием графовых моделей алгоритма является простым, но действенным средством для развития абстрактного мышления учащихся, развития их математических способностей.

Являясь частью дискретной математики, теория графов используется в программировании для создания эффективных алгоритмов.

Предметом первых задач теории графов были конфигурации, состоящие из точек и соединяющих их линий. Такие схемы или диаграммы часто называются *графами*. Точки в конфигурации являются *вершинами* графа, а соединяющие их линии – *рёбрами* или *дугами* графа.



а) б) в) г) д)

Рисунок 1.

Схемы, изображенные на рисунке 1, могут представлять, например, состояние турнира четырех шахматистов. Участники турнира обозначены точками, которые можно пометить цифрами или буквами, а линии, соединяющие пары точек, обозначают соответствующие партии между шахматистами. Рисунок 1а) – состояние перед началом турнира, то есть жеребьевка; рисунки 1б) - 1г) – состоялось три тура; рисунок 1д) – сыграны все 6 партий.

При большом числе вершин и рёбер рисунок графа теряет наглядность. В таких случаях для задания графов и работы с ними используются таблицы специального вида, называемые матрицами. Матричная модель графа используется в работе с графами на ЭВМ. При решении задач используются четыре основных способа представления графа: матрица смежности, матрица инциденций, списки связи и перечни рёбер (чаще для взвешенного графа).

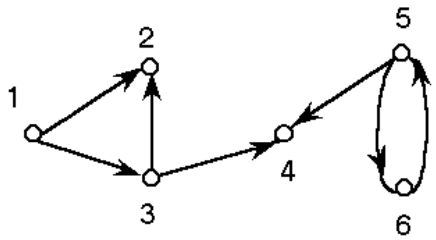
Матрицей смежности графа называют квадратную матрицу, элементы которой равны:

$$a_{ij} = \begin{cases} 0, \textit{ если вершины } i \textit{ и } j \textit{ не смежные} \\ 1, \textit{ если вершины } i \textit{ и } j \textit{ смежные} \end{cases}$$

Матрицей инцидентности называется матрица размера m x n (m – количество ребер, n – количество вершин), элементы которой равны:

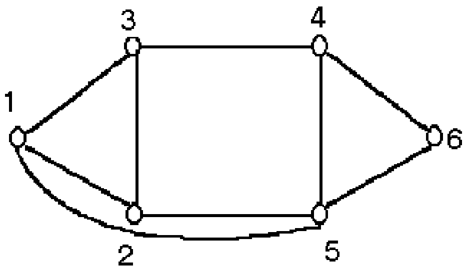
$$a_{ij} = \begin{cases} 0, \textit{ если ребро } a_i \textit{ не инцидентно вершине } A_j \\ 1, \textit{ если ребро } a_i \textit{ инцидентно вершине } A_j \end{cases}$$

Пример 1. Матрица инцидентности ориентированного графа.



$$\begin{array}{c|ccccccc|}
 & \langle 1,2\rangle & \langle 1,3\rangle & \langle 3,2\rangle & \langle 3,4\rangle & \langle 5,4\rangle & \langle 5,6\rangle & \langle 6,5\rangle \\
\hline
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{array}$$

Пример 2. Матрица инцидентности неориентированного графа.



$$\begin{array}{c|ccccccccc|}
 & \{1,2\} & \{1,3\} & \{1,5\} & \{2,3\} & \{2,5\} & \{3,4\} & \{4,5\} & \{4,6\} & \{5,6\} \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
5 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}$$

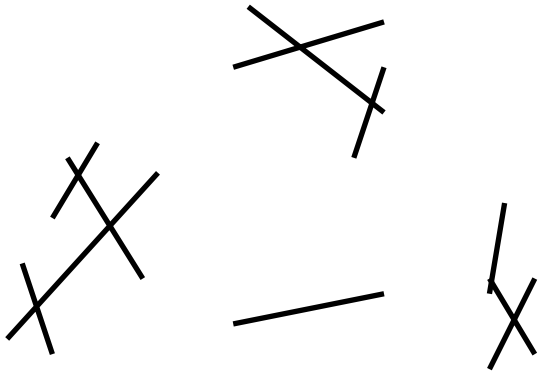
Более удобным способом представления графа является матрица *смежности*. Основным преимуществом матрицы смежности является тот факт, что за один шаг можно получить ответ на вопрос типа «существует ли ребро из x в y?».

Матрицы смежности для графов, приведённых в примерах 1 и 2, имеют вид:

$$\begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
\qquad
\begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
2 & 1 & 0 & 1 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 1 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

Пример 1. Иголки. На столе разбросаны иголки, образующие несколько кучек. Две иголки считаются принадлежащими одной кучке, если они имеют хотя бы одну общую точку. Определить количество кучек, образованных иголками, и количество иголок в каждой кучке. [3] с 89.

Данная задача является примером прикладной задачи геометрического характера, при решении которой можно эффективно использовать алгоритмы на графах.



Решение: Во входном файле дано число N – количество иголок и в следующих N строчках указаны координаты концов каждой иголки. Считываем координаты концов каждой иголки в массивы X1, Y1, X2, Y2 (либо в двумерный массив A[1..n,1..4]), организуем попарный перебор всех иголочек для определения, пересекается ли данная пара иголочек:

```
for i:=1 to n-1 do
 for j:=i+1 to n do
If Peres_li(X1[I],Y1[I],X2[I],Y2[I],X1[J],Y1[J],X2[J],Y2[J])
 Then a[i,j]:=1 Else a[i,j]:=0;
```

(определяем, пересекаются ли два отрезка, или хотя бы соприкасаются, формируем матрицу смежности a[nxn], где a[i,j]=1, если иголки с номерами i и j пересекаются, и 0 – в противном случае.

В данной задаче используем модель графа: в качестве вершин графа выбираются иголки, а в качестве ребер графа – их пересечение: если пара соответствующих иголок пересекается, то эти две иголочки считаем смежными, соединяем их ребром и таким образом строим матрицу смежности графа. По матрице смежности строится матрица достижимости графа. Вершину A_j называют *достижимой* из A_i, если существует путь из вершины A_i в вершину A_j.
Процедура построения матрицы достижимости графа:

```
procedure Reach;
  var   s,t:set of 1..10;i,j,l:integer;
  begin
   for i:=1 to n do
    begin
     t:=[i];
     repeat
       s:=t;
       for l:=1 to n do
        if l in s then
         for j:=1 to n do
          if a[l,j]=1 then t:=t+[j];
     until s=t;
     for j:=1 to n do
       if j in t then r[i,j]:=1;
    end;
end;
```

Матрица смежности приведённого графа может иметь вид:

0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0

Матрица достижимости примет вид:

1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1

По матрице достижимости графа определяем количество кучек (количество изолированных областей в графе) и количество иголок в каждой из них.

Например, если заданы исходные данные задачи по представленному рисунку, то в качестве результата должно быть выдано: количество кучек – 4, количество иголок в каждой кучке – 5 3 1 3.

Пример 2. Game. В некоторой игре одно двузначное число можно заменить на другое по следующему правилу: любая из двух цифр исходного числа заменяется на сумму или разность его цифр (в случае разности из большей цифры вычитается меньшая). Для двузначных чисел a и b построить последовательность чисел минимальной длины, начинающуюся с числа a, заканчивающуюся числом b, а каждое следующее число в цепочке можно получить из предыдущего по указанному выше правилу или определить, что это сделать невозможно (Костанайская городская олимпиада 2004г)

Пример: для чисел 12 31 получаем цепочку чисел 12 32 31

Указание: Один из оптимальных вариантов решения задачи: построение графовой модели и применение метода просмотра вершин графа в ширину – даёт цепочку чисел минимальной длины. Строится матрица смежности, в которой двузначные числа 10..99 являются вершинами, а в качестве ребер – отношение между числами u v: можно ли число v получить из числа u по указанному правилу:

```
   For i:=10 To 99 Do
      { формируем матрицу смежности для чисел от 10..99 }
     Begin
         e:=i Mod 10; d:=i Div 10;
         l:=Abs(e-d);
                     s:=e+d;
         If s<10 Then
           Begin
             s1:=s*10+e;s2:=d*10+s;
             If i<>s1 Then a[i,s1]:=1;
             If i<>s2 Then a[i,s2]:=1;
           End;
          If l>0 Then
            Begin
              s3:=l*10+e;s4:=d*10+l;
              If i<>s3 Then a[i,s3]:=1;
              If i<>s4 Then a[i,s4]:=1;
            End;
    End;
     {используем метод просмотра вершин графа в ширину}    u:=u1;yk1:=1;yk2:=1;oger[yk1]:=u;
log[u]:=1;predok[u]:=0;
 While yk1<=yk2 Do
   Begin
     u:=oger[yk1];
     For v:=10 To 99 Do
       If (a[u,v]=1)and(log[v]=0)Then
         Begin
           k2:=yk2+1;oger[yk2]:=v;
           log[v]:=1;predok[v]:=u
         End;
         yk1:=yk1+1;
    End;
```

В качестве результата восстанавливаем из очереди цепочку минимальной длины чисел заданной последовательности, используя массив предков:

```
Z:=v1;
Repeat
  Write(z);
  Z:=predok(z);
Until z=0;
```

СПИСОК ЛИТЕРАТУРЫ:

1. А.Б. Даулеткулов. Олимпиады по информатике. – Алматы, РНПЦ «Дарын», 1999г.
2. Окулов С.М. Программирование в алгоритмах. – М.:БИНОМ. Лаборатория знаний, 2002. – 341с
3. Цыганова А.Д. Алгоритмы на графах. Учебное пособие. – Костанай, 2007. – 104 с.