```
tr(M, it) {
   r += it->second;
   // (*it).first == [string], (*it).second == [int]
}
```

Как и в случае с `set`, элементы `map` хранятся упорядоченными по ключу. Поэтому не следует при работе с `map::iterator` модифицировать `it->first`: если вы нарушите правила упорядочивания элементов в `map`, за последствия никто отвечать не возьмётся.

В остальном контейнер `map` по интерфейсу практически эквивалентен контейнеру `set`.

Также важно помнить, что `operator []` при обращении к несуществующему элементу в `map` создаст его. Новый элемент при этом будет инициализирован нулём (либо конструктором по умолчанию, если это не тривиальный тип данных). Данная особенность `map` может быть удобной, потому как выполнять операции с элементами можно не задумываясь об их присутствии в `map`. Существенным моментом является то, что `operator []` не является константным (то есть может изменить объект, для которого вызван), поэтому им нельзя пользоваться, если `map` передан как const reference. Используйте `map::find(element)`:

```
void f(const map<string, int>& M) {
   if(M["the meaning"] == 42) {
     // Так нельзя! M передан как const reference
   }
   if(M.find("the meaning") != M.end() &&
                   M.find("the meaning")->second == 42) {
      // А можно именно так
      cout << "Don't Panic!" << endl;
   }
}
```

Литература и ссылки

1. *Scott Meyers*, Effective STL. Addison Wesley Professional, 2001.
2. `http://www.sgi.com`
3. `http://msdn.microsoft.com`

Теория графов: определения и задачи

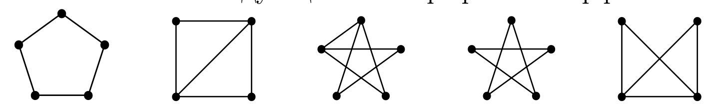
Ю. Кудряшов, П. Митричев

▷ **Определение 1.** *Графом* [6] называется пара $\Gamma = (V, E)$ из конечного множества *вершин* V и множества *рёбер* E, элементами которого являются (неупорядоченные) пары вершин графа Γ.

Граф можно представлять себе как множество точек, некоторые пары которых соединены линиями.

▷ **Определение 2.** Графы Γ_1 и Γ_2 называются *изоморфными*, если существует такая биекция $f\colon V(\Gamma_1) \to V(\Gamma_2)$, что вершины A и B графа Γ_1 соединены ребром тогда и только тогда, когда вершины $f(A)$ и $f(B)$ соединены ребром в графе Γ_2.

Задача 1. Какие из следующих пяти графов изоморфны?



Задача 2. Нарисуйте все неизоморфные друг другу графы с не более чем четырьмя вершинами.

Задача 3. Нарисуйте граф, вершинами которого являются натуральные числа от 1 до 15, а рёбрами соединены числа, одно из которых делится на другое.

Задача 4. а) Постройте граф с пятью вершинами, в котором нет ни трёх попарно соединённых, ни трёх попарно несоединённых вершин.

б) Докажите, что в каждой компании из шести человек найдутся либо три попарно знакомых, либо три попарно незнакомых человека.

Задача 5. Пусть в некоторой компании среди любых трёх человек найдутся два друга. Обязательно ли эту компанию можно разбить на две группы, так что всякие два человека из одной группы — друзья?

Задача 6. Найти наибольшее возможное количество рёбер в графе с n вершинами, если известно, что среди произвольных а) трёх, б*) четырёх его вершин есть две, не соединённые ребром.

▷ **Определение 3.** *Степенью вершины* A называется число выходящих из неё рёбер. Обозначение: $\deg A$.

[6] Точнее, *неориентированным графом без петель и кратных рёбер.*

Задача 7. Укажите степени всех вершин графов из задач 1, 2 и 3.

Задача 8. Докажите, что в графе с более чем одной вершиной есть две вершины одинаковой степени.

Задача 9. Докажите, что сумма степеней вершин произвольного графа равна удвоенному количеству его рёбер.

▷ **Определение 4.** *Путём* в графе называется конечная последовательность вершин (не обязательно различных), в которой всякие две соседние вершины соединены ребром. Путь называется *проходящим по данному ребру*, если это ребро соединяет некоторую пару соседних вершин пути. *Циклом* называется путь, в котором первая и последняя вершины совпадают. *Длиной пути* называется число рёбер, по которым проходит этот путь.

▷ **Определение 5.** Граф Г называется *гамильтоновым*, если в нём существует путь, содержащий каждую вершину ровно один раз.

Задача 10. Докажите, что графы додекаэдра и икосаэдра гамильтоновы.

▷ **Определение 6.** Граф называется *связным*, если для любых двух его вершин существует путь, начинающийся в первой из них и заканчивающийся во второй.

Задача 11. Какие из графов задач 1, 2 и 3 связны?

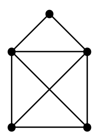
▷ **Определение 7.** Связный граф называется *деревом*, если в нём не сущесвует цикла, все рёбра которого различны.

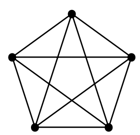
Задача 12. Докажите, что в любом дереве есть вершина степени 1.

Задача 13. Докажите, что в дереве число вершин на 1 больше числа рёбер.

▷ **Определение 8.** Граф называется *эйлеровым*, если в нём существует цикл, проходящий по каждому ребру ровно один раз.

Задача 14. Доказать, что следующие графы эйлеровы:





Задача 15. Докажите, что граф эйлеров тогда и только тогда, когда он связен и степень каждой его вершины чётна.

Задача 16*. В турнире без ничьих участвовало n команд. Каждая команда сыграла с каждой ровно по одному разу. Докажите, что можно так занумеровать команды числами $1, \ldots, n$, что $(i+1)$-я команда выиграла у i-й (для произвольного $i = 1, \ldots, n-1$).

Задача 17*. (*теорема Рамсея*) а) Докажите, что для произвольных натуральных m, n существует натуральное k такое, что в произвольном графе с k вершинами найдется либо m попарно соединенных ребрами вершин, либо n попарно несоединенных. Наименьшее такое k обозначается $R(m, n)$.

б) Найдите $R(3, 4)$.

Задача 18*. Докажите, что если для произвольных двух различных вершин A и B графа с n вершинами $\deg A + \deg B \geqslant n$, то этот граф гамильтонов (см. задачу 10).

▷ **Определение 9.** Назовём *расстоянием* между вершинами связного графа наименьшую длину пути, соединяющего эти вершины. *Диаметром графа* называется наибольшее расстояние между его вершинами.

▷ **Определение 10.** Граф называется *регулярным* графом *валентности* k, если степень каждой его вершины равна k.

▷ **Определение 11.** *Графом Мура* называется регулярный граф валентности k диаметра не больше 2, число вершин которого равно $k^2 + 1$.

Задача 19*. а) Докажите, что в регулярном графе валентности k и диаметра 2 не может быть больше, чем $k^2 + 1$ вершина.

б) Приведите примеры графов Мура при $k = 1, 2, 3$.

в) Существует ли граф Мура при $k = 7$?

г) Существует ли граф Мура при $k = 57$?

д) Докажите, что ни при каких других значениях k не существует графов Мура.

Задача 20*. Дан правильный 50-угольник. В одной из его вершин стоит доктор Фауст. У него есть три возможности: 1) бесплатно перейти в диаметрально противоположную точку; 2) заплатив Мефистофелю 1 рубль 05 копеек, перейти на соседнюю вершину против часовой стрелки; 3) получив от Мефистофеля 1 рубль 05 копеек перейти на соседнюю вершину по часовой стрелке. Известно, что доктор Фауст побывал в каждой вершине (хотя бы один раз). Докажите, что на каком-то отрезке пути кто-то кому-то заплатил не меньше 25 рублей.

Задача 21. В турнире по олимпийской системе участвовали n команд. Сколько всего было сыграно матчей?

Задача 22. Докажите, что граф с n вершинами, степень каждой из которых не менее $\frac{n-1}{2}$, связен.

Задача 23. В связном графе все вершины имеют степень 100. Докажите, что после удаления любого из рёбер он остаётся связным.

Задача 24. Докажите, что из любого связного графа можно выкинуть вершину и выходящие из неё рёбра так, чтобы он остался связным.

Задача 25*. В кубической коробке $n \times n \times n$ лежали n^3 единичных кубиков. Кубики высыпали, каждый просверлили по диагонали, затем все плотно нанизали на нить и связали в кольцо (соединили вершину первого кубика с вершиной последнего). При каких n получившееся «ожерелье» можно убрать обратно в коробку?

Задача 26*. Все 28 Петиных одноклассников имеют по различному числу друзей в этом классе. Сколько из них дружат с Петей? А если одноклассников n?

Задача 27*. (Теорема Кэли) В графе с n вершинами каждая вершина соединена с каждой ребром (такой граф называется *полным*). Докажите, что существует ровно n^{n-2} способов выкинуть несколько рёбер так, чтобы оставшийся граф был деревом.

▷ **Определение 12.** *Ориентированным графом* называется граф, на рёбрах которого поставлены стрелки. Его рёбра называются *дугами.* Более формально, ориентированный граф — это пара $\Gamma = (V, E)$ из конечного множества вершин V и множества дуг E, элементами которого являются упорядоченные пары вершин графа Γ.

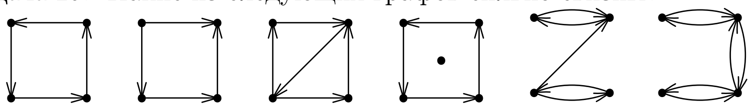
Заметим, что у нас в ориентированном графе разрешаются дуги из вершины в себя саму (*петли*), несколько дуг из одной вершины в другую (*кратные дуги*), «встречные» дуги (из A в B и из B в A).

То, что раньше называлось графом, мы теперь будем называть *неориентированным графом.*

Задача 28. Дайте (формальные!) определения *пути* и *цикла* в ориентированном графе.

▷ **Определение 13.** Ориентированный граф называется *сильно связным*, если для любых двух его вершин существует путь как из первой во вторую, так и из второй в первую.

Задача 29. Какие из следующих графов сильно связны?



▷ **Определение 14.** Ориентированный граф называется *связным*, если он окажется связным неориентированным графом после того, как мы сотрём стрелки с его дуг.

Задача 30. а) Приведите пример связного, но не сильно связного ориентированного графа.

б) Приведите пример связного ориентированного графа, в котором для некоторых двух вершин A и B нет пути ни из A в B, ни из B в A.

Задача 31. Докажите, что на рёбрах связного неориентированного графа можно так расставить стрелки, чтобы из одной из вершин существовали пути во все остальные.

Задача 32. Можно ли так расставить на рёбрах полного неориентированного графа стрелки, чтобы в полученном ориентированном графе не было циклов?

Задача 33. В полном неориентированном графе на рёбрах как-то расставили стрелки. Докажите, что найдётся вершина, из которой существуют пути во все остальные.

Задача 34*. В полном неориентированном графе с не менее чем тремя вершинами на рёбрах как-то расставили стрелки. Докажите, что можно заменить не более одной дуги на противоположную так, чтобы полученный граф стал сильно связным.

▷ **Определение 15.** Количество дуг, входящих в вершину, называется *входной полустепенью* (или *полустепенью захода*) этой вершины. Количество выходящих дуг называется *выходной полустепенью* (или *полустепенью исхода*).

Задача 35. Найдите входные и выходные полустепени каждой вершины для всех графов из задачи 29.

Задача 36. Что можно сказать о сумме всех входных полустепеней и сумме всех выходных полустепеней одного и того же ориентированного графа?

Задача 37. Сформулируйте и докажите критерий эйлеровости ориентированного графа.

Задача 38. (Цикл де Брюина) Для того, чтобы открыть кодовый замок (с кнопками от 0 до 9), необходимо набрать код из четырёх цифр, причём не важно, что было нажато до набора правильного кода. За какое наименьшее количество нажатий его можно гарантированно открыть?

Задача 39. 20 школьников решали 20 задач. Каждый решил ровно две задачи, и каждую задачу решили ровно двое. Доказать, что можно устроить разбор задач так, чтобы каждый рассказал одну решённую им задачу.

▷ **Определение 16.** Граф называется *двудольным*, если его вершины можно разбить на две группы (называемые *долями*) так, чтобы все рёбра (или дуги) были между различными долями.

Паросочетанием называется такой набор рёбер графа, что каждая вершина графа является концом не более одного ребра из набора. Паросочетание называется *совершенным*, если каждая вершина является концом ровно одного ребра паросочетания.

Раскраска вершин графа называется *правильной*, если никакие две вершины одного цвета на соединены ребром. Граф называется *k-дольным*, если правильная раскраска его вершин возможна k цветами и не менее.

Задача 40. Какие графы из задач 1, 2 и 3 являются двудольными? А сколько доль в остальных?

Задача 41. Равносильна ли двудольность неориентированного графа отсутствию циклов нечётной длины?

Задача 42. Любое ли дерево двудольно?

Задача 43*. (Теорема Холла) В некой компании n юношей. При каждом $k = 1, 2, \ldots, n$ для любых k юношей в этой компании найдется не менее k девушек, знакомых хотя бы с одним из рассматриваемых k юношей. Можно ли сосватать всех юношей за знакомых девушек? Является ли это условие необходимым?

Иными словами, верно ли, что в двудольном неориентированном графе (с n вершинами в первой доле) существует паросочетание размера n тогда и только тогда, когда для каждого набора из k вершин первой доли с ними соединены хотя бы k вершин второй доли?

Задача 44*. (Обобщение задачи 39) Докажите, что в любом регулярном двудольном неориентированном графе есть совершенное паросочетание.

Задача 45*. Верно ли, что при любой правильной раскраске k-дольного неориентированного графа в k цветов найдётся путь из k разноцветных вершин?

▷ **Определение 17.** Граф называется *планарным*, если его можно нарисовать на плоскости, изобразив вершины точками, а рёбра (или дуги) — непересекающимися кривыми. Граф, который нарисован на плоскости указанным выше образом, называется *плоским*. Части, на которые плоский граф делит плоскость (включая внешнюю часть) называются его *гранями*.

Задача 46. Какие графы из задач 1, 2 и 3 являются планарными?

Задача 47. (Формула Эйлера) Пусть в неориентированном плоском связном графе B вершин, P рёбер и Γ граней. Тогда $B + \Gamma - P = 2$.

Задача 48. Чему равно $B + \Gamma - P$ для несвязного неориентированного плоского графа?

Задача 49. Пусть B, P и Γ — количества вершин, рёбер и граней многогранника соответственно. Чему равно $B + \Gamma - P$?

Задача 50. а) Докажите, что в плоском неориентированном графе $2P \geqslant 3\Gamma$.

б) Докажите, что в плоском двудольном неориентированном графе $P \geqslant 2\Gamma$.

Задача 51. Докажите, что следующие графы не планарны:

а) Полный неориентированный граф с 5 вершинами. Этот граф обозначается K_5.

б) Двудольный неориентированный граф с 3 вершинами в первой доле и 3 вершинами во второй доле, причём каждая вершина первой доли соединена с каждой вершиной второй (такой граф называется *полным двудольным*). Этот граф обозначается $K_{3,3}$.

в) Произвольный неориентированный граф, у которого степени всех вершин не меньше шести.

▷ **Определение 18.** *Подграфом* данного графа называется граф, который получается из данного выкидыванием некоторых вершин и рёбер (дуг).

Два неориентированных графа называются *гомеоморфными*, если один можно получить из другого следующими операциями: взять ребро и добавить посередине этого ребра вершину; взять вершину степени 2 и заменить её и выходящие из неё рёбра на одно ребро (заметим, что эти две операции взаимно обратны).

Задача 52*. (Теорема Понтрягина-Куратовского) Докажите, что неориентированный граф планарен тогда и только тогда, когда у него нет подграфа, гомеоморфного K_5 или $K_{3,3}$.

Алгоритмы на графах (семинар)

В. Матюхин

Семинар — это форма теоретического занятия, когда школьникам предлагаются различные задачи на доказательство каких-либо свойств разобранных алгоритмов, на их модификацию и т. д. Все это позволяет лучше понять как сами алгоритмы, так и области их применения и особенности реализации.

Предлагаемый семинар по графам разбит на несколько разделов. В каждом разделе сначала приводятся задачи, а затем обсуждаются их решения. Будет полезно, если перед тем, как читать разборы задач, читатель некоторое время подумает над задачами и попытается их решить самостоятельно.

Предполагается, что читателю знакомы алгоритмы обхода графа в ширину и глубину, алгоритмы поиска кратчайших путей (Дейкстры, Флойда, Форда–Белмана) и построения каркасов минимального веса (Краскала и Прима).

Раздел 1. Обход в глубину и в ширину

В задачах этого раздела предполагается, что граф связный и неориентированный.

1. Докажите, что процедура обхода *в глубину* обойдет все вершины графа (т. е. не может оказаться так, что мы не побывали в некоторой вершине).

2. Докажите, что процедура обхода *в ширину* обойдет все вершины графа.

3. Докажите, что в результате обхода (как в глубину, так и в ширину) те ребра, по которым мы проходим, образуют остовное дерево.

4. В результате обхода в глубину получается дерево. Можно рассмотреть это дерево как корневое (корнем является вершина, из которой мы начали делать обход). Есть ребра исходного графа, которые в это дерево не попали. Докажите, что все такие ребра идут из более глубокой вершины какой-либо ветви построенного дерева в менее глубокую вершину той же ветви, и не бывает ребер, соединяющих вершины, принадлежащие разным ветвям.

5. С помощью алгоритма обхода в ширину можно искать кратчайшие пути в невзвешенном графе. Обоснуйте, почему найденные пути являются кратчайшими.

Обсуждение задач раздела 1

Задача 1 очень проста и интуитивно понятна. Давайте все-таки попробуем этот факт доказать строго. Предположим, что утверждение не верно. Т. е. в какую-то вершину мы в процессе обхода не попали (пусть это вершина B). Тогда, так как граф связен, то из исходной вершины A (из которой мы запускаем алгоритм обхода) существует путь в вершину B. Рассмотрим этот путь. В вершине A наш обход побывал, а в вершине B — нет. Тогда в этом пути есть две соседние (т. е. соединенные ребром) вершины (v_1 и v_2) такие, что в первой из них обход побывал, а во второй — нет. Вот тут-то мы и получаем противоречие. Раз обход побывал в вершине v_1, то, обрабатывая эту вершину, мы перебирали всех ее соседей. В том числе и вершину v_2. А так как мы не были в вершине v_2 ранее (вспомните, по нашему предположению, в вершине v_2 мы вообще не были!), то мы должны были пойти в вершину v_2. А значит, в вершине v_2 мы бы все-таки побывали.

Решение **задачи 2** полностью аналогично.

Задача 3. Давайте чуть-чуть переформулируем утверждение, которое нужно доказать. Во-первых, как мы уже доказали выше, мы побываем во всех вершинах, а значит, ребра, по которым мы пройдем в процессе обхода, образуют связный граф. Таким образом, нам осталось доказать, что в этом графе не будет циклов. Это можно сделать по-разному. Например, предположить, что цикл есть и доказать, что тогда в процессе обхода в какую-то вершину мы приходили дважды, получив тем самым противоречие. Но есть очень изящный способ.

Заметим, что каждый раз, когда мы проходим по ребру, мы добавляем новую вершину в список пройденных. Всего в процессе обхода мы добавим к списку пройденных все вершины графа (кроме начальной — она считается пройденной изначально). А значит, мы добавим $N-1$ вершину (если считать, что всего в графе N вершин). Следовательно, мы пройдем по $N-1$ ребру. Осталось вспомнить, что связный граф из N вершин, в котором $N-1$ ребро — дерево.

Задача 4. Для решения этой задачи нужно хорошо понимать, как устроен алгоритм обхода в глубину. Предположим, что ребро, соединяющее две разные ветви дерева, есть (пусть его концы — вершины u и v). Пусть, для определенности, обход в глубину сначала побывал в вершине u, а затем — в вершине v. Причем, поскольку эти вершины по нашему предположению принадлежат разным поддеревьям, то в вершину v обход попал, когда полностью закончил рассмотрение вершины u. Тогда, перебирая в вершине u всех ее соседей, обход в глубину увидел бы вершину v (ведь они соединены ребром) и должен был бы пойти в v

(ведь мы там еще не были!), но не пошел (по нашему предположению). Противоречие.

Задача 5. Изначально мы побывали только в начальной вершине (до нее расстояние 0). Затем мы добавили всех ее соседей (расстояние до них равно 1). Затем — их соседей (вершины на расстоянии 2) и т. д. Почему для всех вершин найденное расстояние будет кратчайшим? Действительно, пусть до некоторой вершины мы нашли не кратчайшее расстояние. Рассмотрим все такие вершины и выберем наиболее близкую к начальной вершине (назовем эту вершину v). Пусть реальное расстояние до нее равно x (соответственно, когда мы нашли расстояние, оно оказалось больше, чем x). Тогда у этой вершины есть сосед (обозначим его u), до которого расстояние $x-1$, причем это расстояние найдено правильно. Тогда, рассматривая эту вершину, мы из нее пойдем в вершину v (так как в вершине v мы еще не были) и запишем в вершину v число x. Противоречие. Почему, когда мы просматриваем u, вершина v еще не просмотрена? Потому что, как отмечено вначале, мы сначала просмотрим все вершины с расстоянием 0, потом — с расстоянием 1, 2, 3 и т. д. (подумайте, почему). А, значит, когда мы будем просматривать вершину u, расстояние до которой равно $x-1$, вершину, из которой мы пришли в v по нашему предположению мы еще не просмотрим (так как, раз в v мы по предположению запишем число больше x, то расстояние до вершины, из которой мы в нее придем, будет x или больше).

Раздел 2. Алгоритмы Дейкстры, Флойда и Форда-Белмана

В этом разделе все графы предполагаются ориентированными (рассматривать неориентированные графы с отрицательными весами ребер неинтересно).

1. Как с помощью алгоритма Флойда проверить, есть ли в графе циклы отрицательного веса? Как это же проверить с помощью алгоритма Форда-Белмана?

2. Как для двух вершин a и b установить, существует ли *кратчайший* путь из a в b?

3. Постройте пример графа (с отрицательными весами ребер), в котором путь, найденный алгоритмом Дейкстры, не будет кратчайшим.

4. Можно ли модифицировать алгоритм Дейкстры, чтобы искать длины путей в графах с отрицательными весами ребер, следующим образом? Найдем сначала самое маленькое ребро в графе (пусть оно имеет вес $-a$). Теперь прибавим к весу каждого ребра число $(a+1)$. После этого все ребра будут иметь положительный вес. Теперь с помощью алгоритма Дейкстры найдем кратчайший путь в новом графе, и будем считать, что он же является кратчайшим в исходном.

5. Рассмотрим следующую задачу. Пусть каждому ребру приписан вес. Пусть стоимость пути вычисляется как произведение весов ребер пути. Можно ли применять для подсчета кратчайшего пути в этом случае алгоритмы Дейкстры, Флойда, Форда-Белмана: а) если веса всех ребер положительны? б) если веса ребер неотрицательны? в) если веса ребер произвольные?

Обсуждение задач раздела 2

Первые две задачи этого раздела являются скорее контрольными вопросами или упражнениями, в то время как задачи 3, 4 и 5 требуют довольно серьезных размышлений.

Задача 1. Начнем с алгоритма Флойда. Что такое цикл отрицательного веса? Это путь из вершины в саму себя. А значит, нужно лишь просмотреть диагональ построенной алгоритмом Флойда матрицы и проверить, есть ли на ней отрицательные числа (подумайте, почему если цикл отрицательного веса существует, он будет найден алгоритмом Флойда).

С алгоритмом Форда-Белмана чуть сложнее. Во-первых, если окажется, что цикл отрицательного веса недостижим из той вершины, которая является начальной для алгоритма, то мы никак не узнаем о его существовании. Если же цикл отрицательного веса есть, то узнать об этом довольно просто. Напомним, что алгоритм Форда-Белмана состоит из $N-1$ шага, на каждом из которых мы перебираем все ребра и обновляем те значения расстояний до вершин, которые удается улучшить. Сделаем еще один аналогичный (N-й) шаг. Оказывается, что если на этом шаге меняется хотя бы одно из значений, то в графе есть цикл отрицательного веса, если нет — то циклов отрицательного веса, достижимых из начальной вершины, нет. Действительно, если в графе нет цикла отрицательного веса, то любой кратчайший путь из начальной вершины куда-либо состоит не более, чем из $N-1$ ребра, и, следовательно, будет найден за $N-1$ шаг алгоритма. При дальнейших шагах значения меняться уже не будут. Если же в графе есть цикл отрицательного веса, то нам выгодно «бегать» по этому циклу сколь угодно долго, и от этого расстояния до каких-то вершин будут уменьшаться. А значит, такое уменьшение какого-либо из значений произойдет и на N-м шаге (ведь если на каком-то шаге ни одно значение не меняется, то, из-за того, что шаги делаются одинаково, дальше уже ничего никогда меняться не будет).

Задача 2. Формулировка задачи кажется довольно странной. В то же время, задача отнюдь не так тривиальна, как кажется на первый взгляд. Итак, что же нужно проверить, чтобы с уверенностью уметь сказать, существует ли кратчайший путь между двумя вершинами или