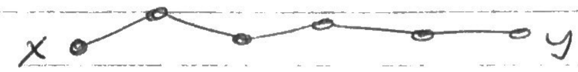
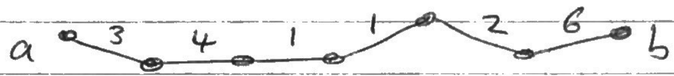
10.6 Shortest path problems

The length of a path in a simple graph is just the number of edges



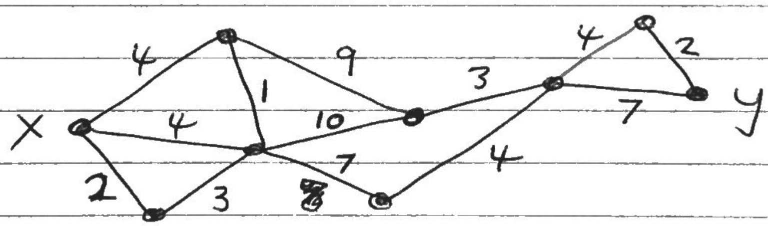
path from x to y of length 5.

In a weighted graph each edge has a number (weight) and the length of a path is now the sum of its edge weights



weighted path from a to b of length 17.

For example, the weights could represent miles on a road network between towns:



What is the shortest length path from X to y? Looks like 21 miles.

We want a simple and efficient algorithm to find the shortest length path between any two vertices in a connected simple weighted graph.

You could imagine ants setting off in every direction along edges from X, taking k seconds to cross an edge of weight k. The first ant to reach y above would show the shortest path. But this would be a complicated algorithm.

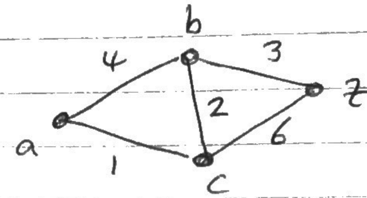
<u>Dijkstra's Algorithm</u> from 1959 gives an elegant solution as follows. We label the vertices in a weighted graph G by $a = v_0, v_1, v_2, \ldots, v_n = z$ where we want to know the shortest path from a to z. Use $w(v_i, v_j)$ for the weight of the edge between vertex v_i and vertex v_j. We assume that $w(v_i, v_j) > 0$ and let $w(v_i, v_j) = \infty$ if there is no edge.

```
Procedure Dijkstra( G graph with vertices
                      a = v_0, v_1, ..., v_n = z
                      and weights w(v_i, v_j) )

for i := 1 to n
      L(v_i) = ∞                 } initialization
L(a) := 0
 S := ∅

While z ∉ S
      u := vertex not in S with L(u) minimal
      S := S ∪ {u}
      for all vertices v not in S
            L(v) := minimum( L(v), L(u) + w(u,v) )
                                         ↖ update labels
return L(z)

{Output is the length of shortest path from
                                     a to z}.
```

Note that this algorithm gives the shortest length. Can easily adapt to get shortest path.

Example

G



The main idea of the algorithm is that paths are examined in a set of vertices S that expands out from vertex a in the while loop. The labels $L(v)$ keep track of the shortest path from a to v so far.

With the above graph G we can see how the algorithm works in a table

iteration	S	$L(a)$	$L(b)$	$L(c)$	$L(z)$
0	$\emptyset$	0	∞	∞	∞
1	$\{a\}$	0	4	1	∞
2	$\{a,c\}$	0	3	1	7
3	$\{a,b,c\}$	0	3	1	6
4	$\{a,b,c,z\}$				

In the first iteration $L(a)$ is minimal and gets added to S. Then we update the labels

$$L(b) = \min(\infty, 0+4) = 4, \quad L(c) = \min(\infty, 0+1) = 1$$
$$L(z) = \min(\infty, \infty) = \infty$$

Now we see that vertex c has the smallest label (of the vertices not in S) and gets added to S in iteration two. The labels for b and z are updated.

The iterations continue until $z \in S$. Then $L(z) = 6$ is returned as the shortest path length.

(can skip)

Proof that Dijkstra's algorithm is correct.

We need a complicated induction argument. Let $P(n)$ be the statement that after the nth step in the algorithm we have

(A) For all $v \in S$, $L(v)$ gives the length of the shortest path from a to v,

(B) For all $v \notin S$, $L(v)$ gives the length of the shortest path from a to v using only vertices in S.

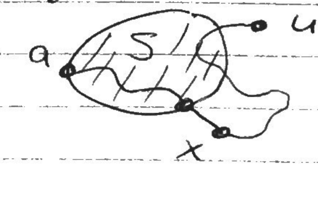
The basis case $P(0)$ is true because $S = \emptyset$ and $L(a) = 0$ while $L(v) = \infty$ for $v \neq a$.

Now we want to prove the inductive step. We assume that $P(k)$ is true and want to show $P(k+1)$ is true where vertex u gets added to S for $L(u)$ minimal and for all $v \notin S$ the labels are updated:

$$L(v) = \min\left(L(v), L(u) + w(u,v)\right).$$

To show that (A) is true for $P(k+1)$ we need to prove that $L(u)$ is the length of the shortest path from a to u.

- If this shortest path only uses vertices from S then $L(u)$ gives shortest length by $P(k)$ part (B).
- If this shortest path uses vertices outside S then let x be the first:



But then $L(x) < L(u)$ and this contradicts $L(u)$ being minimal.

Either way, we have proved (A) for $P(k+1)$.

To show that (B) is true for $P(k+1)$, we look for the shortest path from a to each $v \notin S$ using only the vertices in S and u.

This shortest length = shortest of

paths from a to $y \in S$ $+ w(y,v)$
or
paths for a to u $+ w(u,v)$.

But the length of the shortest path from a to $y \in S + w(y,v)$ is $L(v)$ by $P(k)$ part (B).

And the length of the shortest path from a to u is $L(u)$ as we just proved (A) for $P(k+1)$.

Therefore the length of the shortest path from a to $v \notin S$ using only vertices in S and u is

$$\min\big(L(v),\ L(u) + w(u,v)\big).$$

This is exactly our updated label, so we have proved part (B) for $P(k+1)$.

This completes the inductive step and by induction the algorithm is correct.

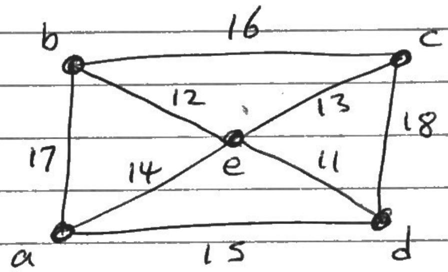
$\square$

Navigation apps would need to use a similer algorithm to find shortest routes.

A similer problem is the famous <u>Traveling Salesperson problem</u>.

This asks for the shortest length Hamilton circuit in a weighted graph.

Example



Solve the Traveling Salesperson problem for this graph.

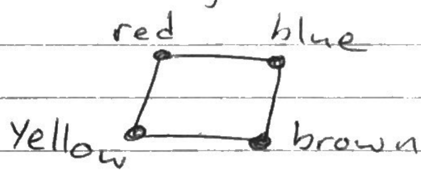
Solution: We are looking for the shortest circuit that passes through every vertex exactly once.
Looks like e, c, b, a, d, e gives the shortest length circuit with 72.

No efficient algorithm is known to solve this problem exactly, though efficient approximation methods are known.
See p714 – 716 in the text.

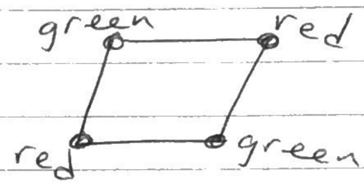
10.8 Graph Coloring

Definition: A coloring of a simple graph is a way of giving every vertex a color so that adjacent vertices have different colors.

Examples: Colorings of C_4



or



Definition: The smallest number of colors needed to color a graph G is called the chromatic number of G. Notation $\chi(G)$.
"chi"

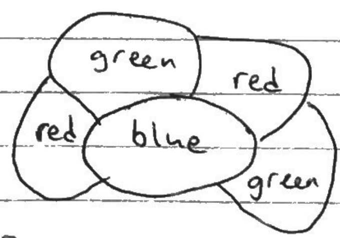
So we see $\chi(C_4) = 2$ and $\chi(C_3) = 3$

Also, remember from an earlier section, that G is bipartite if and only if $\chi(G) \leq 2$.

How big can a chromatic number be?

Answer - very big. Since all vertices in K_n are adjacent we must have $\chi(K_n) = n$ for all $n \geq 1$. (K_n is the complete graph on n vertices.)

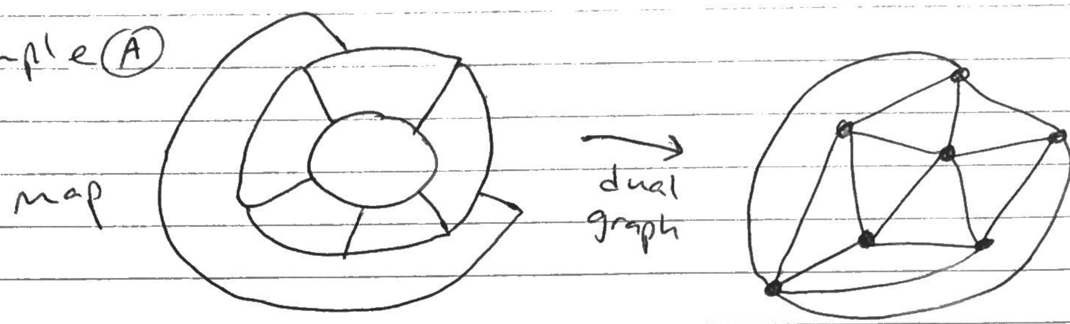
Coloring Maps



A closely related question asks for the smallest number of colors to color a map. Countries (or any regions) sharing a border should have different colors.

Given any map we can make its dual graph: the vertices are the regions and regions sharing a border get edges between their vertices.

Example (A)



We see the map on the left in this example needs 4 colors because the chromatic number of its dual graph is 4 (check).

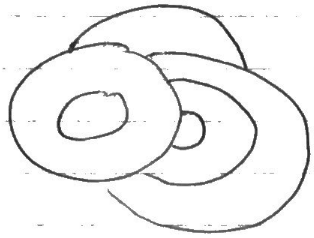
A famous conjecture from 1852 said that any map needed only 4 colors. This was finally proved with the help of a computer in 1976 by Appel and Haken. Their result is

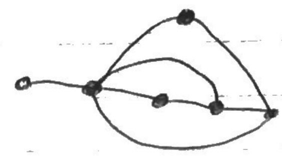
<u>Theorem</u> The chromatic number of a planar graph is ≤ 4.

See the text p 728 for more on this celebrated result.

(Note that a graph is <u>planar</u> if it can be drawn in the plane - or on a sheet of paper - with no edges crossing.)

Example (B) Find the dual graph for this map and find its chromatic number.





Solution. We see the dual graph is

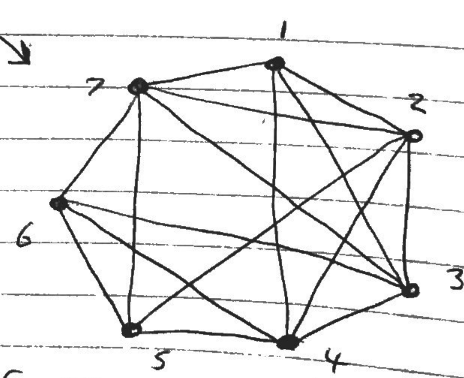
This graph contains triangles (C_3s) so it needs at least 3 colors. It is planar so it doesn't need more than 4. Coloring we see the chromatic number is 3.

Example (C) (see p 731) Suppose finals are to be scheduled for seven courses labeled 1, 2, 3, 4, 5, 6, 7. Also suppose these pairs of courses have common students:

1, 2	2, 3	3, 4	4, 5	5, 6	6, 7
1, 3	2, 4	3, 6	4, 6	5, 7	
1, 4	2, 5	3, 7			
1, 7	2, 7				

How can the finals be scheduled with the smallest number of time slots?

Solution Draw a graph with a vertex for each course. Join vertices by an edge if their courses have a common student.



Then show that the chromatic number of this graph is 4. This means we only need 4 time slots for the finals. Courses with the same color can use the same time