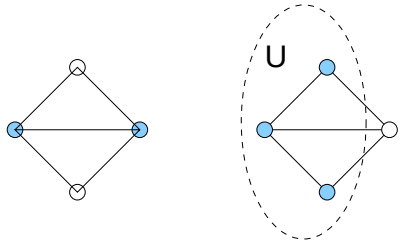
Vertex Cover



The vertex cover U has cardinality $\frac{3}{2}\tau(G)$ (an optimal cover is shown on the left). Hence $R(G,U) = 3/2$. How close to the optimum can we get?

Vertex Cover is an example of a problem for which we can attain some bounded approximation ratio, but this ratio cannot be pushed too close to one.

In the following slides we will present several approximation algorithms for the VC problem.

We will be considering "nearly" four algorithms each of which is based on a distinct idea.

One reason for this overly extensive coverage of the various algorithms is that some of the ideas appear to be extremely novel and may be exportable to other problems.

Moreover, as we will see later, even a small improvement in the best-known approximation ratio for VC will have profound implications. It is curious, therefore, that we have several different algorithms which all achieve the same ratio (asymptotically 2) but there appears to be no way of improving this ratio at the present time.

Simplest Greedy

A natural heuristic for VC is a greedy algorithm which repeatedly picks an edge that has not yet been covered, and places one of its end-points in the current covering set.

GREEDY1 (G)
  $C \leftarrow \emptyset$
  while $E \neq \emptyset$
    Pick any edge $e \in E$ and choose an end-point v of e
    $C \leftarrow C \cup \{v\}$
    $E \leftarrow E \setminus \{e \in E : v \in e\}$
  return C

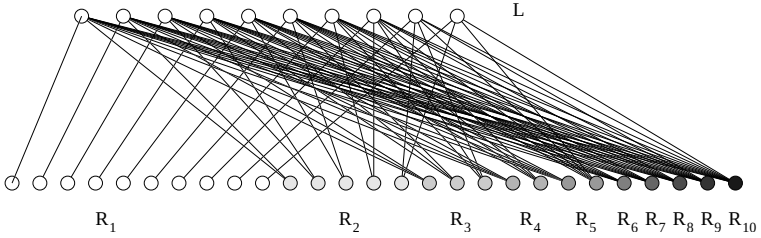
Exercises

- Prove that GREEDY1 always outputs a vertex cover.
- Is the vertex cover problem any easier on a bipartite graph?

How good is this algorithm??

Algorithm analysis

We claim that algorithm GREEDY1 does not achieve any bounded ratio. To see this, consider the following bipartite graph $B = (L, R, E)$. The vertex set L consists of r vertices. The vertex set R is further sub-divided into r sets called $R_1, \ldots, R_r$. Each vertex in R_i has an edge to i vertices in L and no two vertices in R_i have a common neighbour in L; thus, $|R_i| = \lfloor r/i \rfloor$. It follows that each vertex in L has degree at most r and each vertex in R_i has degree i. The total number of vertices $n = \Theta(r \log r)$.



Suppose that (out of sheer bad luck) the algorithm considers an edge out of R_r first, choosing the end-point in R as the vertex to be placed in the cover. Then it picks an edge out of R_{r-1}, again choosing its end-point in R for the cover C; and, so on. Therefore the vertex cover chosen is $C = R$. But L is itself a vertex cover since the graph is bipartite. It follows that the ratio achieved by this algorithm is no better than $|R|/|L| = \Omega(\log n)$.

Clever greedy algorithm

How do we achieve a better ratio than this?

Let us try the obvious strategy of modifying the Algorithm GREEDY1 to be less arbitrary in its choice of vertices to be included in the cover. A natural modification is to repeatedly choose vertices which are incident to the largest number of *currently* uncovered edges.

```
GREEDY2 (G)
    C ← ∅
    while E ≠ ∅
        Pick a vertex v ∈ V of
            maximum degree in the current graph
        C ← C ∪ {v}
        E ← E \ {e ∈ E : v ∈ e}
    return C
```


Algorithm analysis

Let us consider the behaviour of this algorithm on the graph B. It should be easy to see that GREEDY2 could also output R as a vertex cover. It could choose vertices from R_r at the very first stage. After this, it could choose vertices from R_{r-1}. In general, it would choose the highest degree vertices from R at each stage. It is very surprising that a seemingly much more intelligent heuristic does no better than the rather simple-minded heuristic GREEDY1. However this algorithm is not totally useless. It may be shown that it always achieves the ratio $O(\log n)$ for the much more general problem of set cover and hence also for vertex cover.

Maximal matchings and vertex covers

We now describe a different heuristic which achieves a bounded ratio for the vertex cover problem.

The basic idea is to modify GREEDY1 by placing *both* end-points of some uncovered edge into C. Most people find the fact that this algorithm performs better than GREEDY1 and GREEDY2 to be very counter-intuitive at first.

Algorithm MM
Pick any *maximal* matching M in the graph $G = (V, E)$.
Place both end-points of each edge in M into the cover.

Algorithm analysis

Claim. MM always computes a vertex cover in the input graph G. Moreover, it is a 2-approximation algorithm.

Since M is a maximal matching all edges in $E \setminus M$ are such that at least one of their end-points is incident to some $e \in M$ (otherwise, that edge could be added to M to provide a larger matching). Thus every edge in E has at least one end-point in C.

To see that the ratio is 2, consider the edges in M. To cover these edges we need at least $|M|$ vertices, since no two of them share a vertex. This implies that the optimal vertex cover has size at least $|M|$. The cover C contains exactly $2|M|$ vertices.

Exercises

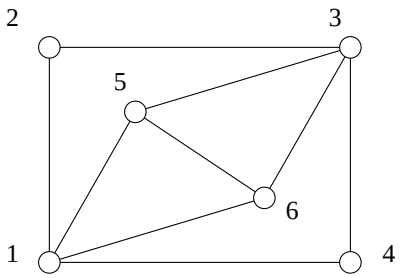
1. What is the behaviour of algorithm MM on the graph B?
2. Show that there exist input graphs for which the performance of MM is no better than a ratio of 2.
3. Show that using a maximum matching instead of a maximal matching does not improve the worst-case performance of MM.

Better algorithms?

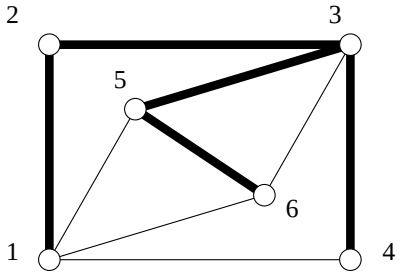
Another algorithm which achieves a ratio of 2 for this problem is due to Savage. This algorithm, which we call DFS, is as simple as the one outlined above. The basic idea is to find a depth-first spanning tree in the graph G. The cover C is then the set of non-leaf nodes in the tree. We leave the analysis of this algorithm as an exercise (proving that the set is a vertex cover is simple, giving a bound on the approximation ratio is non-trivial).

It is an important open problem to find any c-approximation algorithm for the VC problem with $2 - c = \Omega(1)$.

Example



A Depth-first spanning tree



Final vertex cover