EXERCISE 1. Let $G = (V, E)$ be a graph and $w: E \to \mathbb{R}_+$ a weight function. The Laplacian matrix of $G$ and $w$ is the matrix $A: V \times V \to \mathbb{R}$ such that

$$A(u, v) = \begin{cases} \ w(\delta(u)) & \text{if } u = v, \\ -w_{uv} & \text{otherwise}, \end{cases}$$

where $\delta(u) = \{ v \in V : uv \in E \}$ and $w(\delta(u)) = \sum_{e \in \delta(u)} w_e$. Consider again the semidefinite programming relaxation for MAXCUT:

$$\max \ \sum_{uv \in E} \frac{1}{2} (1 - X(u, v))w_{uv}$$

$$X(u, u) = 1 \ \text{for } u \in V,$$

$$X \succeq 0.$$  

(a) Prove that the Laplacian matrix $A$ is positive semidefinite.

(b) Let $\lambda$ be the maximum eigenvalue of $A$. Show that the optimal value of problem (*) is at most $\lambda/2$. Hint: use duality.

(c) An automorphism of $G$ is a permutation $\pi: V \to V$ such that $\pi(u)\pi(v) \in E$ if and only if $uv \in E$ (that is, it is a permutation that preserves the adjacency relation).

A graph is vertex-transitive if for every $u, v \in V$ there is an automorphism $\pi$ of $G$ such that $\pi(u) = v$. Show that, if $G$ is a vertex-transitive graph, then the optimal value of (*) is exactly $\lambda/2$.

(d) Compute the optimal value of (*) when $G$ is a cycle with $n$ vertices and $w(e) = 1$ for all $e \in E$.

EXERCISE 2. Important: You should send the solution to this exercise as a text file by email.

Your program has to work for you to get points. Test it with many instances to be sure it works.

(a) Write a SAGE function that, given a list of real vectors on the $(n-1)$-dimensional unit sphere $S^{n-1}$, uses the random hyperplane method to round them to $\{+1, -1\}$. Your function should have the prototype

$$\text{random_rounding(vector_list)}$$

where vector_list is a list of vectors to be rounded. So for instance one could call:

```python
u = vector(RR, [ 1, 0, 0 ])
v = vector(RR, [ 0.7071, 0, 0.7071 ])
random_rounding([ u, v ])
```

The function should return a list of numbers $\{+1, -1\}$ corresponding to the original vectors.

Note: If $u$ is a SAGE vector (as above), then $u\.degree()$ is its number of components. In this way you can figure out from the list of vectors the dimension of the sphere in which they are given. You may assume all vectors have the same number of components.
(b) Write a SAGE function that, given a SAGE graph and a dictionary containing for each of its edges the weight of the edge, solves the semidefinite programming relaxation of the maximum cut problem.

If $G = (V, E)$ is the graph given to the function, let $X : V \times V \to \mathbb{R}$ be the solution found by the SDP solver. You should compute vectors $x_u \in \mathbb{R}^V$ for $u \in V$ such that $X(u, v) = x_u^T x_v$. Your function should return a dictionary associating with each vertex $u \in V$ the vector $x_u$, and it should also return the objective value of the solution found by the solver.

The prototype of your function should be

```python
solve_maxcut_sdp(G, w)
```

where $G$ is the graph and $w$ is the weight function (given as a dictionary).

Note: Check out the slides about SAGE, where there is an overview of how to use graphs in SAGE and how to use dictionaries.

You can use any SDP solver you want, but you must be able to extract the solution in SAGE yourself. For instance, you could use either CVXOPT, which comes with SAGE, or use the small script I wrote for solving problems with CSDP from SAGE (available from the course website, see also the slides of the 27.04 lecture).

(c) Finally, combine both your functions in a third function that receives a SAGE graph and a dictionary assigning weights to the edges of the graph, finds a cut in the graph with weight at least $0.878c^*$, where $c^*$ is the weight of a maximum-weight cut.

The prototype for your function should be

```python
maxcut_gw(G, w)
```

where $G$ is the SAGE graph and $w$ the dictionary giving weights to the edges.

Your function should simply apply the random hyperplane rounding procedure repeatedly, until a cut of large weight is found. Then it should return the cut as a list of edges, its weight, the optimal value of the semidefinite programming relaxation (to certify that the weight of the cut found is at least $0.878c^*$), and the number of times the randomized rounding procedure was called.

Notice that it is possible that your function will never return, but since this is quite unlikely you should just ignore the possibility.