ALGOLAB 2015

Q&A Session

Michael Hoffmann <hoffmann@inf.ethz.ch>

Contents

- Exam
- Results of the Winter Games
- Discussion of Clues and Sith

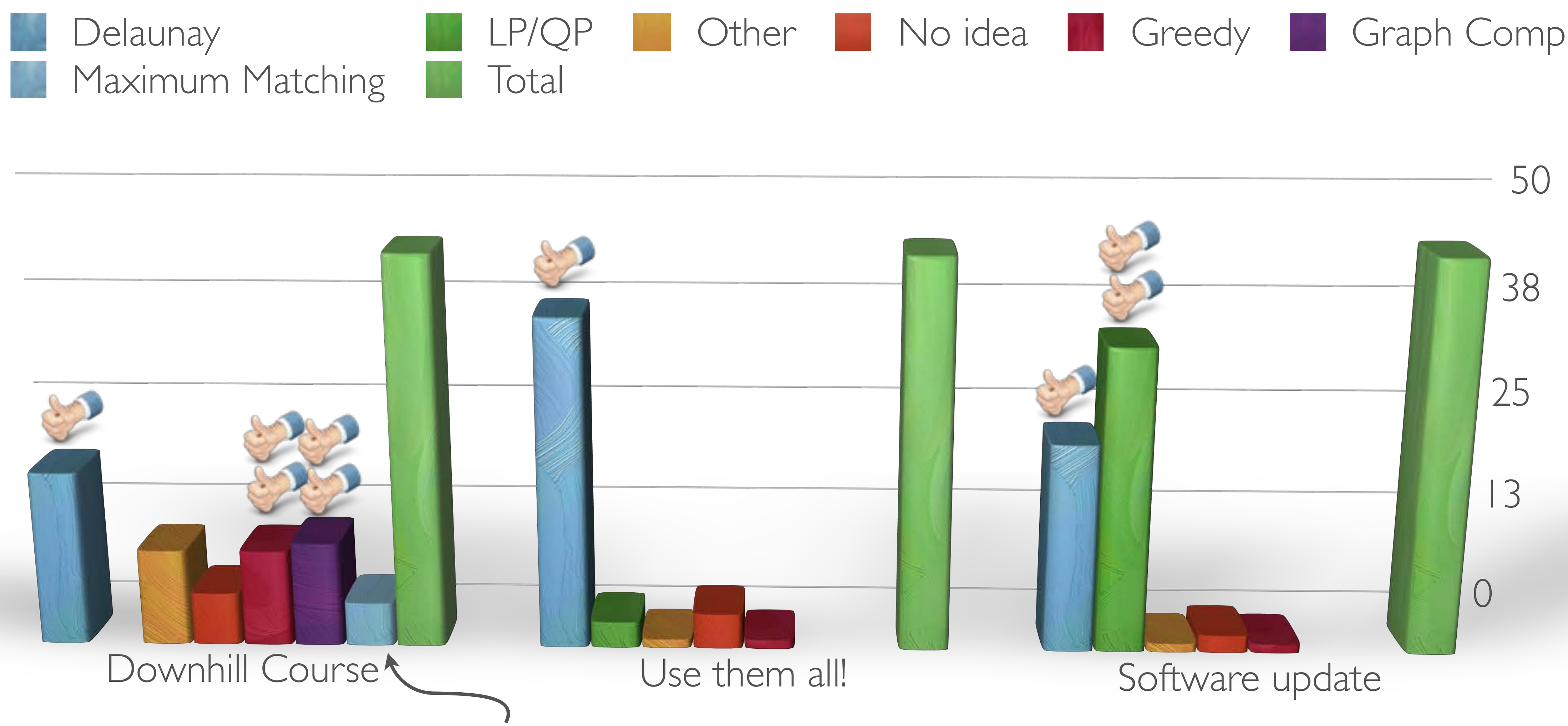
EXAM

What you can expect

- Documentation on `judge.inf.ethz.ch/doc/`
- text editors for programming (emacs, vi, …)
- terminal based compiling
- Tools and IDEs such as eclipse may work or not. (We will make an effort but no guarantees…)

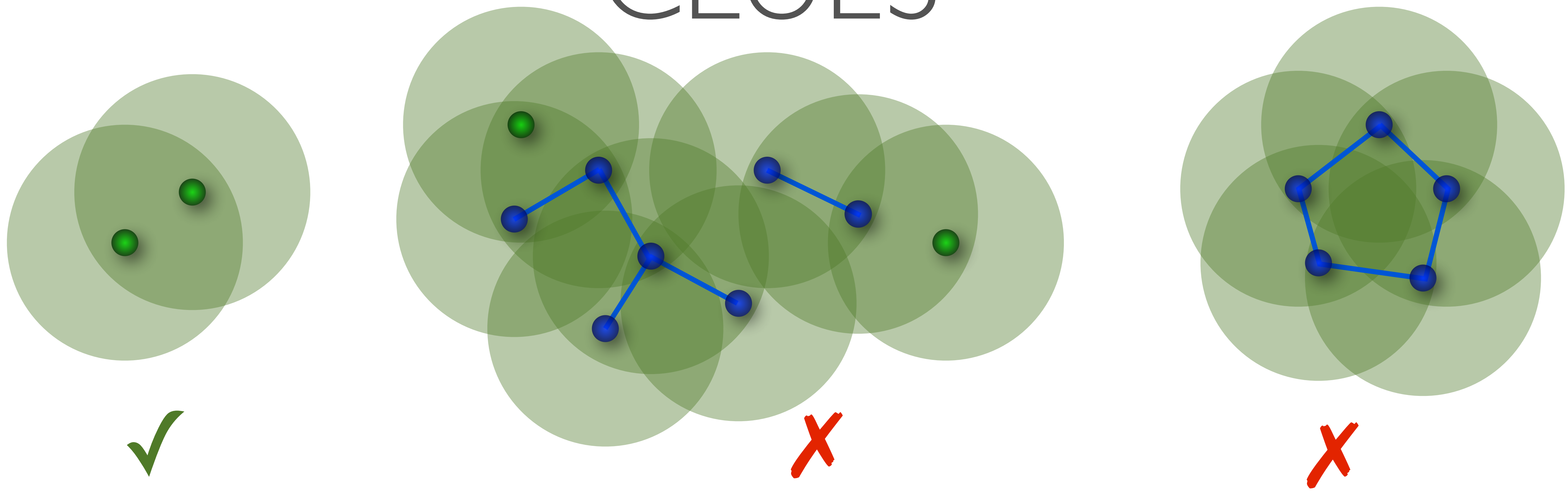
→ Make sure you know how to work in such an environment.

WINTER GAMES



Attention: Kőnig's Theorem holds for bipartite graphs only!
The graph here is not bipartite necessarily.

CLUES

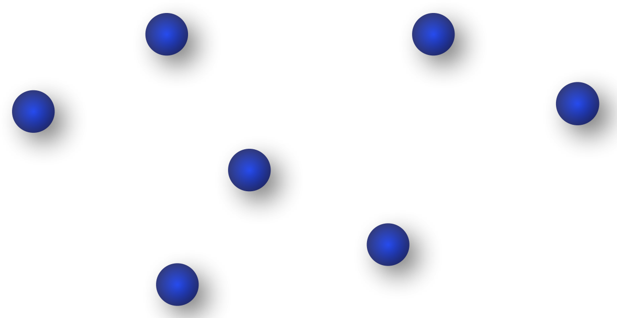


Main aspects:

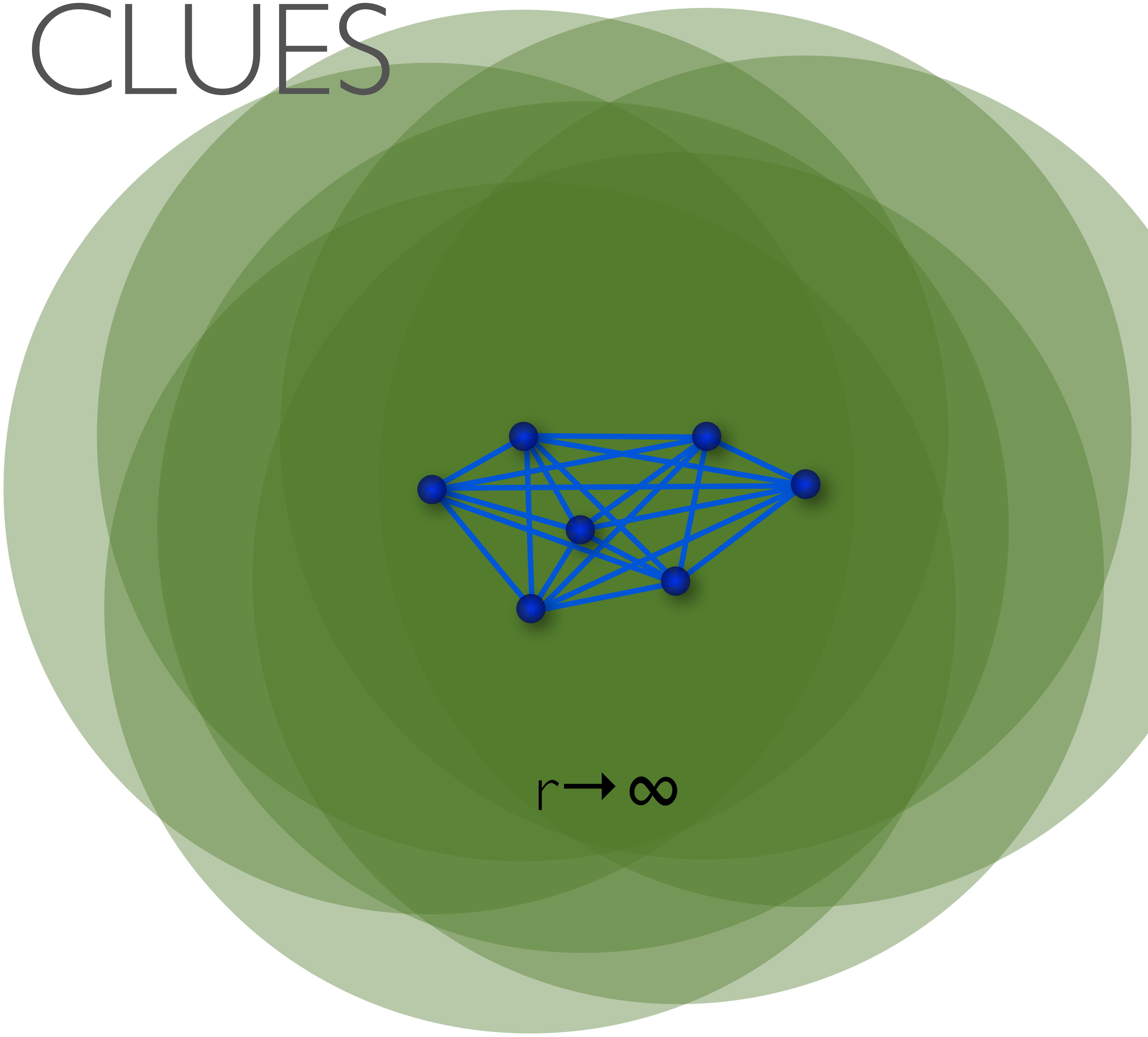
- Communication graph G (given implicitly) — Can be computed brute-force in $O(n^2)$ time.
- Interferences — Classic vertex coloring: Is G bipartite?
- Clue routing — Do the query vertices connect to the same component of G?

Similar in spirit to H1N1 and Olympic Winter Games.

CLUES



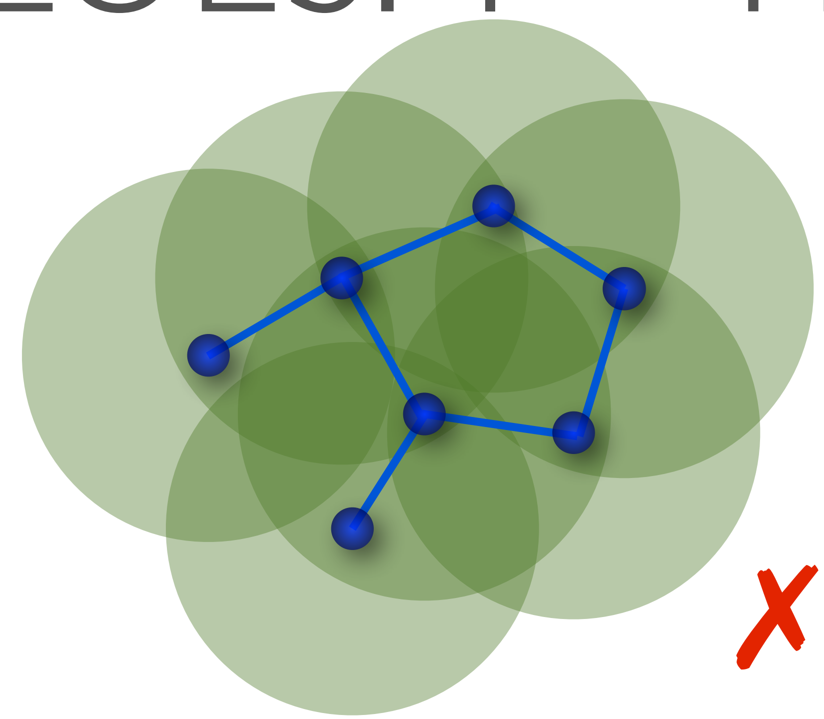
r=0



$r \to \infty$

Communication graph: two extremes

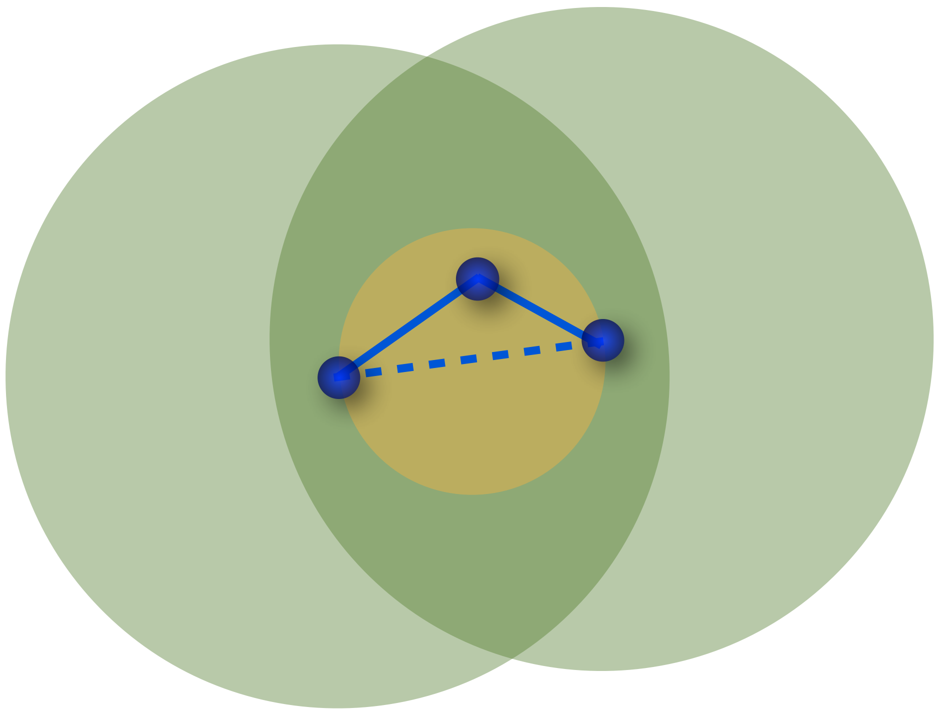
CLUES: 1ST TESTSET



- Compute communication graph brute-force (n≤5’000)
- Test for interferences using BFS/DFS greedy coloring (or using `boost::is_bipartite`)
- No clue routing needed

→ 20 points

CLUES: 2^{ND} + 3^{RD} TESTSETS



Claim. The components of T are the same as the components of G.

Suppose not and consider a shortest edge uv of G s.t. u and v are in different components of G∩D. As uv is not in D, the disk with diameter uv contains another point w. Both uw and vw are strictly shorter than uv and therefore edges in G. As uv is a shortest edge of G whose endpoints are in different components of G∩D, we know that both u and w as well as v and w are in the same component of G∩D.

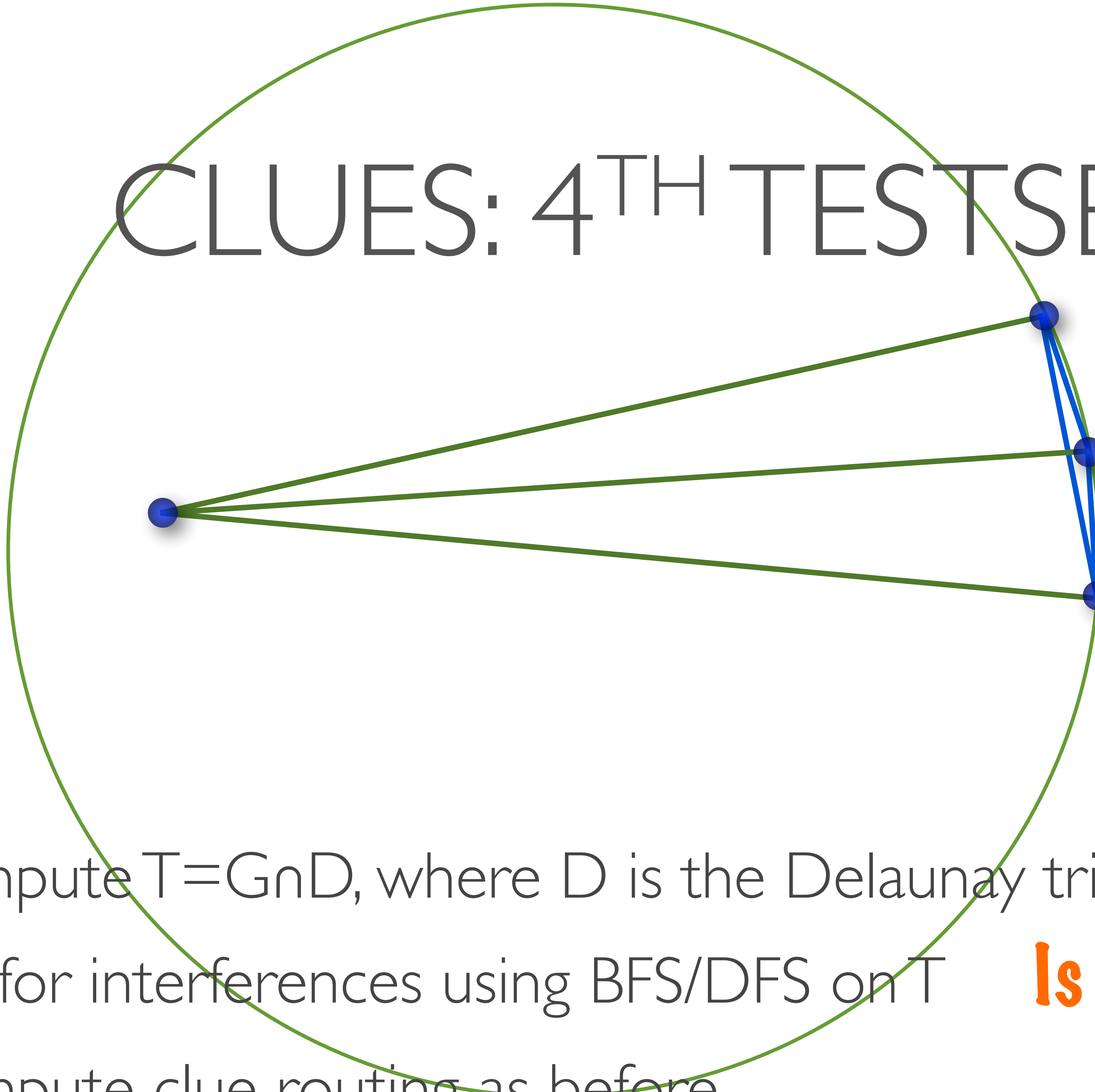
⚡ u and v are in different components of G∩D

- No need to test for interferences
- Bottleneck is building G, can we do better?
- Compute T=G∩D, where D is the Delaunay triangulation of S
- Compute clue routing using BFS/DFS on T in linear time

→ 60 points

For the 3rd testset, precompute the component structure. (For the 2nd testset, there at most 20 clues.)

CLUES: 4TH TESTSET



No!

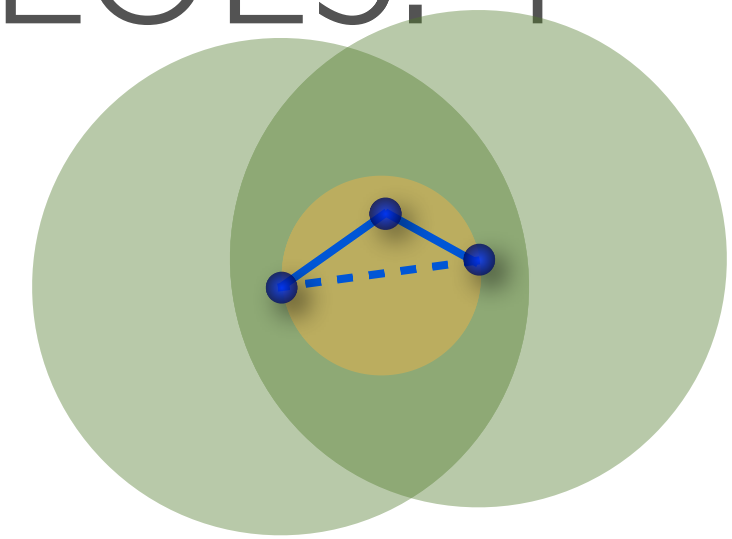
- Compute T=G∩D, where D is the Delaunay triangulation of S
- Test for interferences using BFS/DFS on T

Is this enough?

- Compute clue routing as before

→ 20 points

CLUES: 4TH TESTSET



Consider a shortest missed edge (on an odd cycle in G but not an edge of D)...
=> If there is such an edge, then there is a triangle in G two edges of which we find in T; search for these...

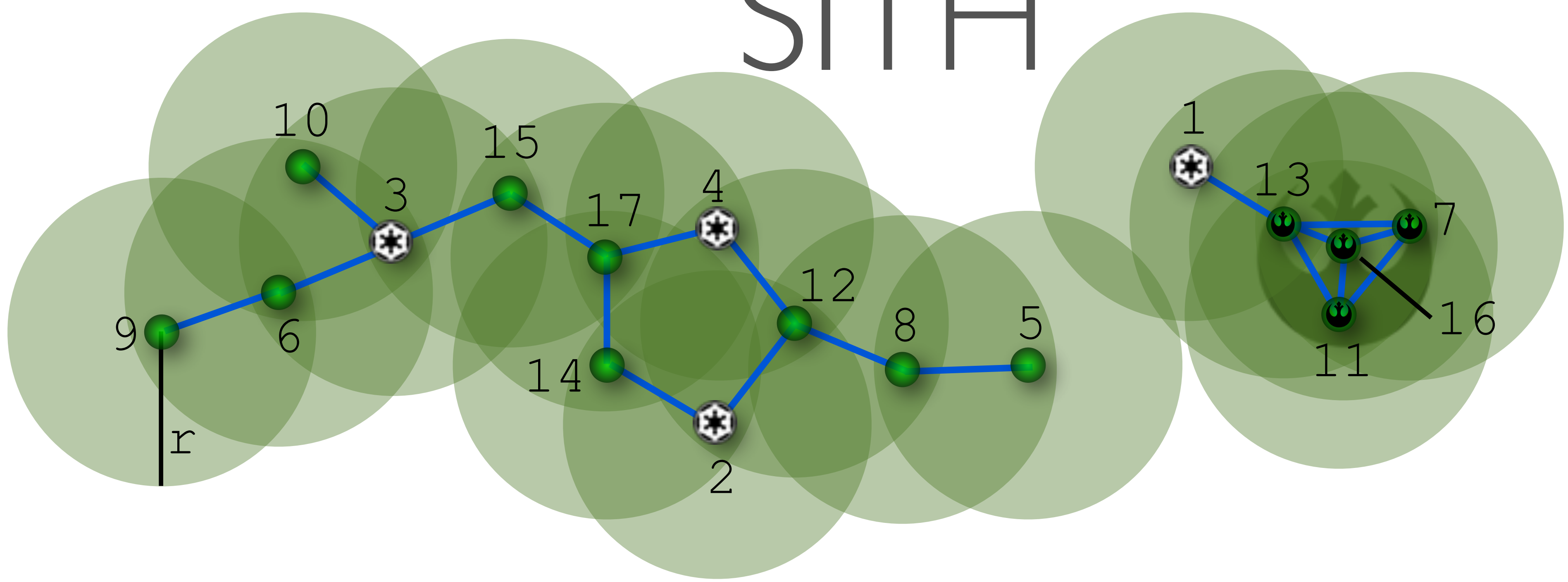
- Compute T=G∩D, where D is the Delaunay triangulation of S
- Test for interferences using BFS/DFS on T
  - If T is not-bipartite then neither is G

    Graypes...
  - Else compute a shortest monochromatic edge **or** for each vertex test whether there is a triangle in its neighborhood.
- Compute clue routing as before

If T has a vertex of degree ≥6, then we are done. Why?...

→ 20 points

SITH



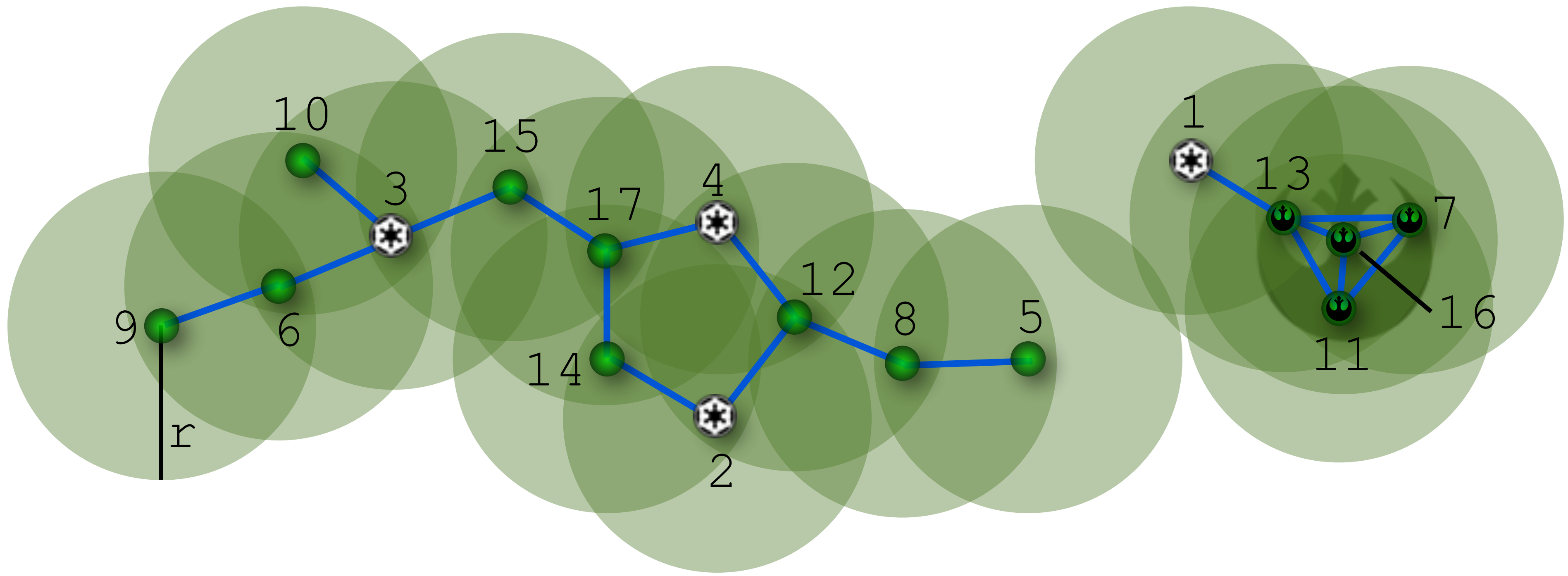
Here $k=4$ is optimal. **Obs.** $k \leq \lfloor n/2 \rfloor$

Problem:

- Given: Sequence $p_0,\ldots,p_{n-1}$ of points and $r>0$.
- Distance induced graph G: p_ip_j connected $\Leftrightarrow$ $\|p_i - p_j\| \leq r$.
- Optimisation: Max. k s.t. $|\text{max. comp. in } G[\{p_k,\ldots,p_{n-1}\}]| \geq k$.

Similar in spirit to H1N1, Clues and Winter Games.

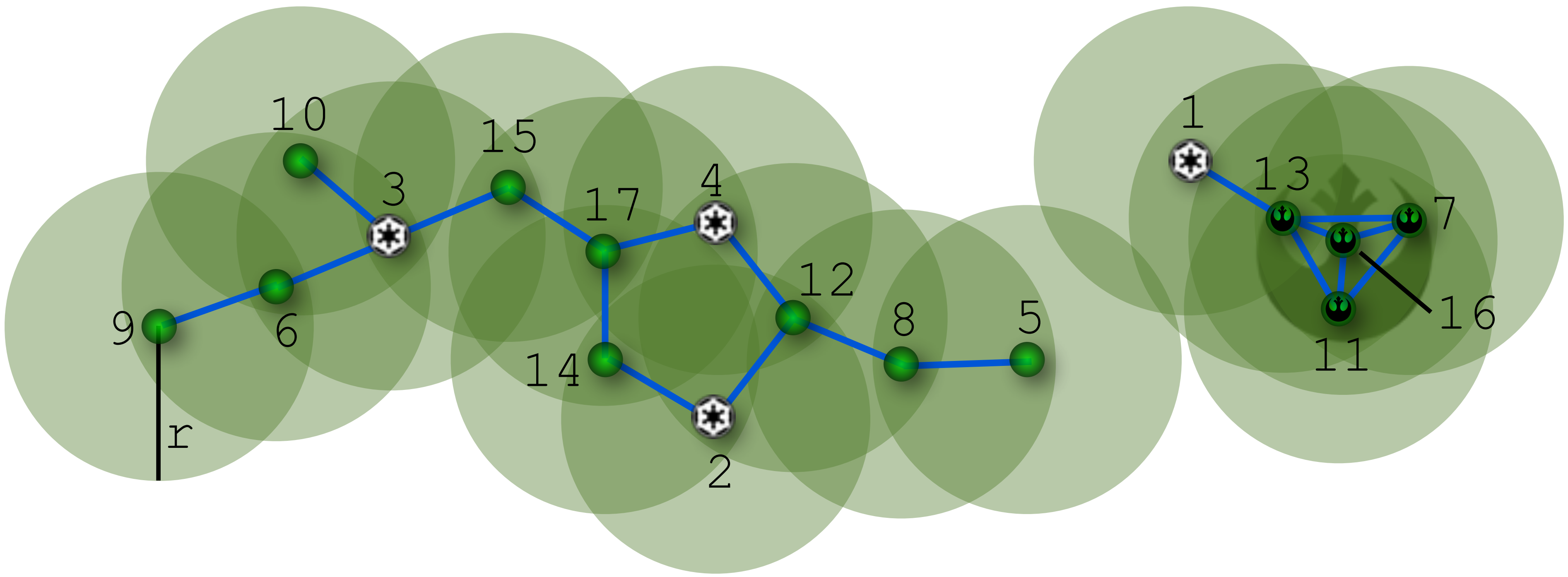
SITH: 1ST TESTSET



- G can be computed brute-force in $O(n^2)$ time.
- Compute components using custom union-find or `boost::connected_components` in $O(n)$ time.
- Linear search for k yields $O(n^3)$ time.

→ 25 points

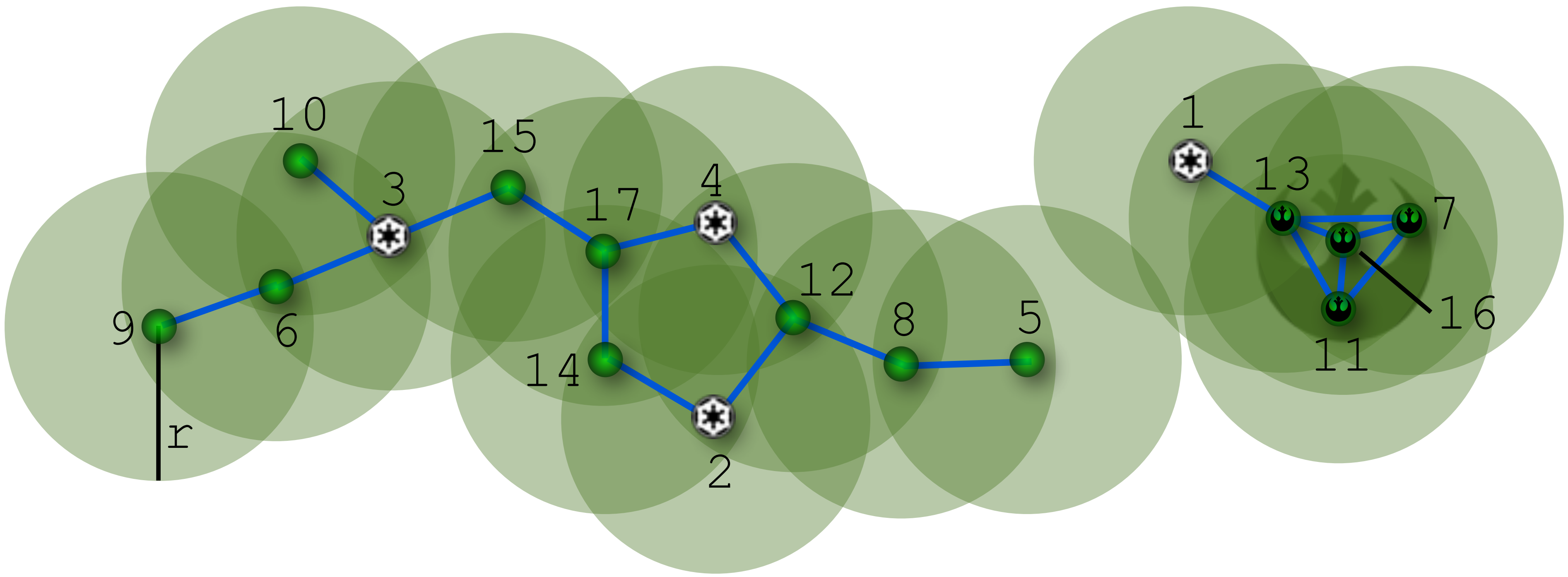
SITH: 2ND TESTSET



- G can be computed brute-force in $O(n^2)$ time.
- Compute components using custom union-find or `boost::connected_components` in $O(n)$ time.
- Binary search for k yields $O(n^2 \log n)$ time.

→ 50 points

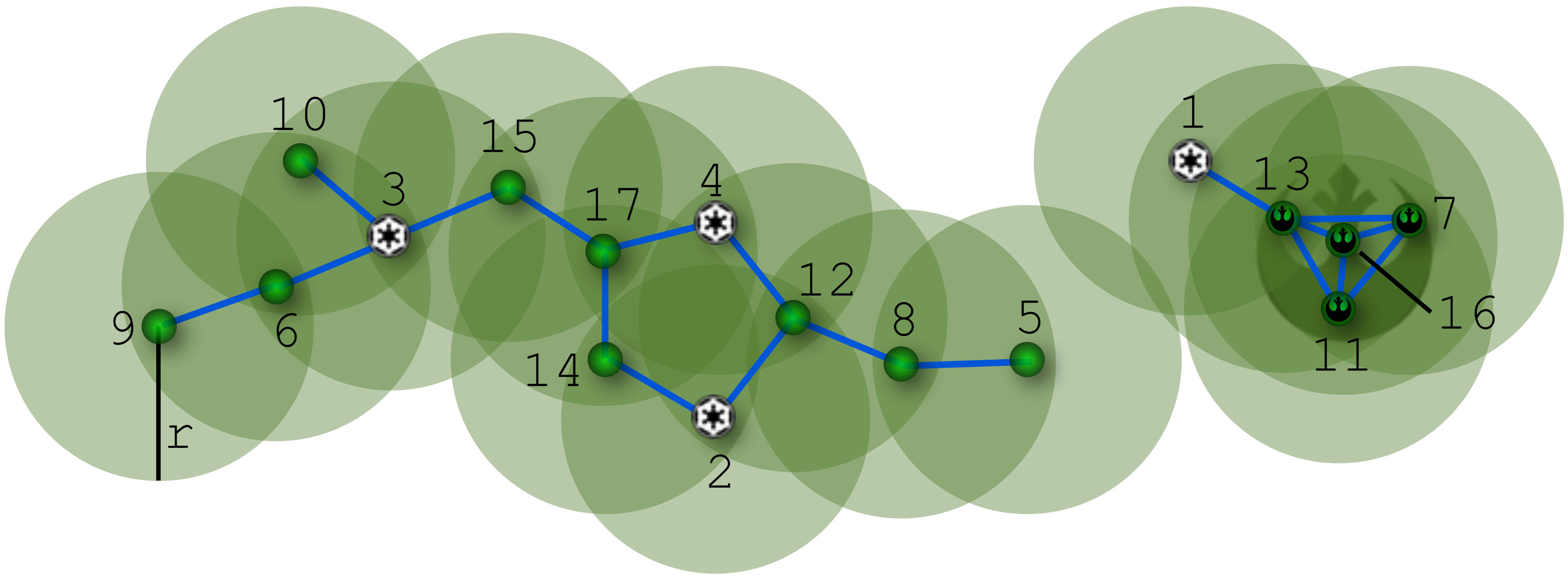
SITH: 3RD TESTSET



- Update both G and its components (playing the sequence backwards) in total $O(n^2)$ time.

→ 75 points

SITH: 4TH TESTSET



- Compute G using Delaunay in O(n log n) time.
- Combined with binary search for k gives O(n $\log^2$ n) time.
- (Update Delaunay rather than recompute: O(n log n).)

→ 100 points