Stamps Solution

Antonis Thomas
Algolab 2015

Stamps - The Problem

- **Given**:
  - stamps $0 \leq s, \ell \leq 200$
  - lamps
  - walls $0 \leq w \leq 2000$

Stamps - The Problem

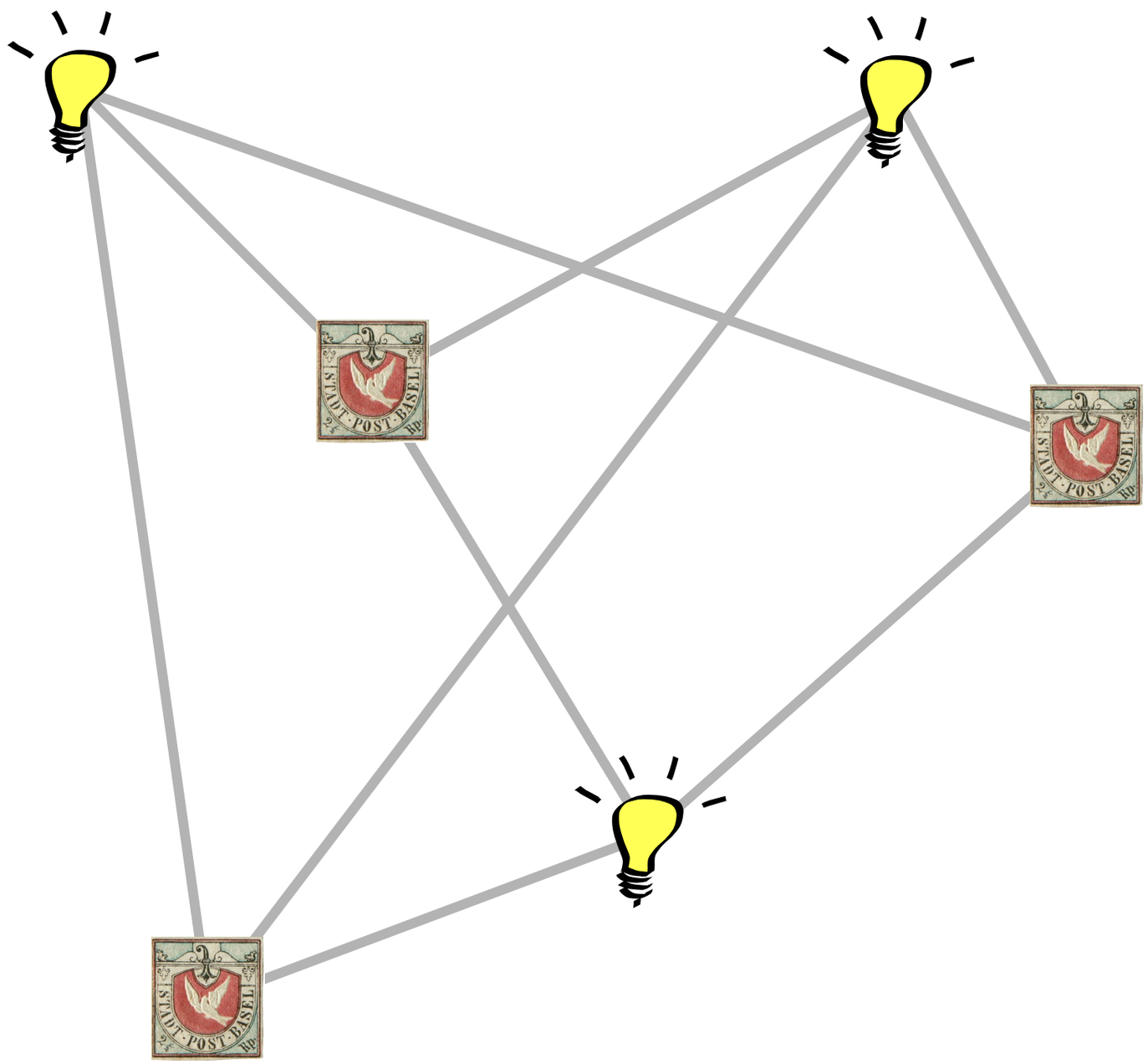
- **Given**:
  - stamps $0 \leq s, \ell \leq 200$
  - lamps
  - walls $0 \leq w \leq 2000$

Each stamp j has a maximum intensity M_j.

Stamps - The Problem

- **Lamp** i:
  - power p_i , illuminates object at distance r with intensity $\frac{p_i}{r^2}$.
- **Wall**:
  - blocks light
- **Requirement**:
  - power p_i of every lamp:
  - total light intensity I_j for every stamp j: $1 \leq I_j \leq M_j$

1st case: No walls!



At this point…

- You should be able to formulate the LP/QP.
  - decide on correct number types.
  - pick the right solver.

Number types

- Integral numbers or not?

Number types

- Integral numbers or not?
- Already for the Input Type …
- $1 \leq I_j \leq M_j$
- where $I_j = \sum_{l \in \{\text{lamps that can see stamp j}\}} \frac{p_l}{d(j,l)^2}$

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/basic.h>
#include <CGAL/QP_models.h>
#include <CGAL/QP_functions.h>
#include <CGAL/Gmpq.h>
#include <vector>

typedef CGAL::Gmpq                                          ET;
typedef CGAL::Quadratic_program<ET>                         Program;
typedef CGAL::Quadratic_program_solution<ET>                Solution;
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
```

```
std::vector<K::Point_2> lamp(200), stamp(200);
std::vector<K::Segment_2> wall(2000);

int main(){
  std::cin.sync_with_stdio(false);
  int t, l, s, w;
  std::cin >> t;


while(t--){
    std::cin >> l >> s >> w;
    Program lp(CGAL::SMALLER, true, 1, true, (1<<12));
    for(int i = 0; i < l; i++) std::cin >> lamp[i];
    for(int i = 0; i < s; i++) {
  int I;
  std::cin >> stamp[i] >> I;
  lp.set_b(2*i, -1); lp.set_b(2*i + 1, I);
}
for(int i = 0; i < w; i++) std::cin >> wall[i];

for(int i = 0; i < s; i++)
  for(int j = 0; j < l; j++){
      double dist2 = CGAL::squared_distance(stamp[i], lamp[j]);
      lp.set_a(j, 2*i, ET(-1)/dist2);
      lp.set_a(j, 2*i+1, ET(1)/dist2);
  }
```

“Nonnegative”

- solve_nonnegative_linear_program:
  - imposes the lower bound $\mathbf{x} \geq 0$
  - ignores any upper bounds

“Nonnegative”

- solve_nonnegative_linear_program:
  - imposes the lower bound $\mathbf{x} \geq 0$
  - ignores any upper bounds

…regardless of what upper and lower bounds you might have set.

“Nonnegative”

- Suggestion:
  - Always first try to solve without _nonnegative_
  - If you get TL and your Program satisifies the conditions mentioned in the previous slide then try also with _nonnegative_
  - (*Most likely* it will not be the case that _nonnegative_ will make a difference)

quadratic vs _linear_

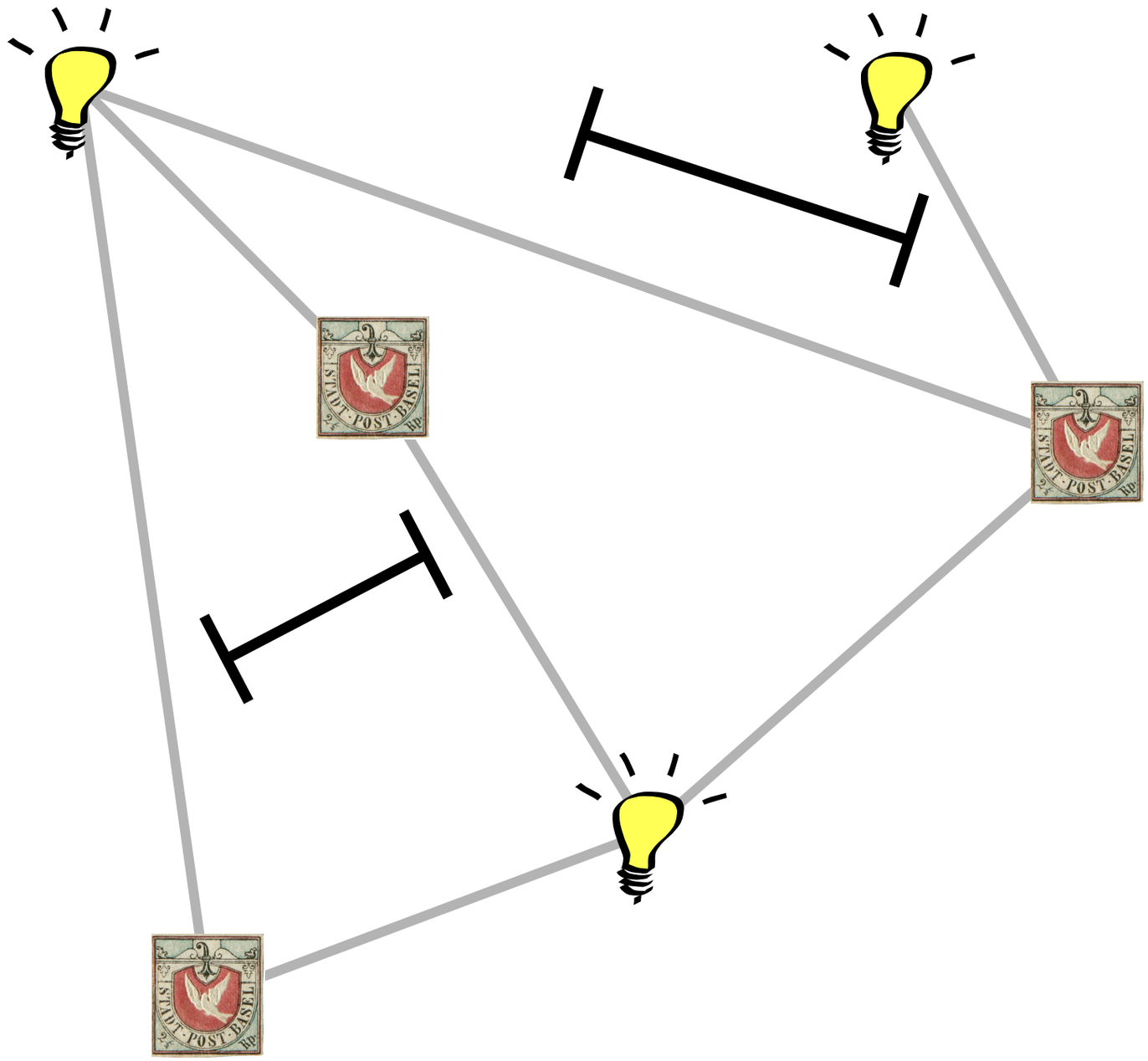
- If you have set up a quadratic program (e.g. Portfolios)

- and then use solve_nonnegative_linear_program

- the quadratic part will be ignored…

```
  Solution s = CGAL::solve_linear_program(lp, ET());
    std::cout << (s.is_infeasible() ? "no" : "yes") << std::endl;
  }
  return 0;
}
```

2nd case: With walls!



How to…

- Add the walls to our program?
- Can’t be that hard..

How to…

- Add the walls to our program?
- Can’t be that hard..
- In this problem, it was enough to just check for every (lamp,stamp) pair if the corresponding segment has an intersection with some wall.

```
bool no_wall_between(const K::Segment_2& seg, const int w){
  for(int i = 0; i < w; i++)
    if(CGAL::do_intersect(seg, wall[i])) return false;
  return true;
}
```

```
bool no_wall_between(const K::Segment_2& seg, const int w){
  for(int i = 0; i < w; i++)
    if(CGAL::do_intersect(seg, wall[i])) return false;
  return true;
}
```

```
for(int i = 0; i < s; i++)
  for(int j = 0; j < l; j++)
    if(no_wall_between(K::Segment_2(stamp[i], lamp[j]), w)){
        double dist2 = CGAL::squared_distance(stamp[i], lamp[j]);
        lp.set_a(j, 2*i, ET(-1)/dist2);
        lp.set_a(j, 2*i+1, ET(1)/dist2);
    }
```