Threefold Problem Set #4
Christmas Presents

December 13, 2017

Fetch the problem sheet at the entrance!

Threefold Problem Set: Christmas Presents

Goal: simulate the thinking steps of a six hour exam in one hour.

First Hour: (17:15 – 18:00)

- 3 problems on paper
- think about them
- sketch your solution on paper
- no coding required

Second Hour: (18:10 –)

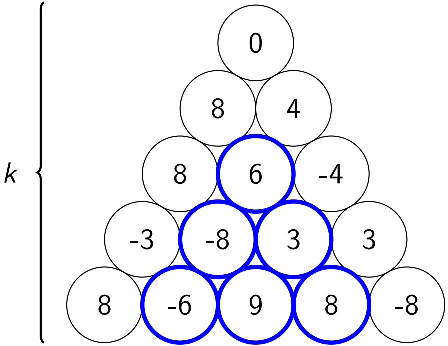
- solution discussion

Afterwards:

- Q&A session

Fetch the problem sheet at the entrance!

Alice's Accumulation – naive solution



Goal: maximize the sum of a sub-pile.

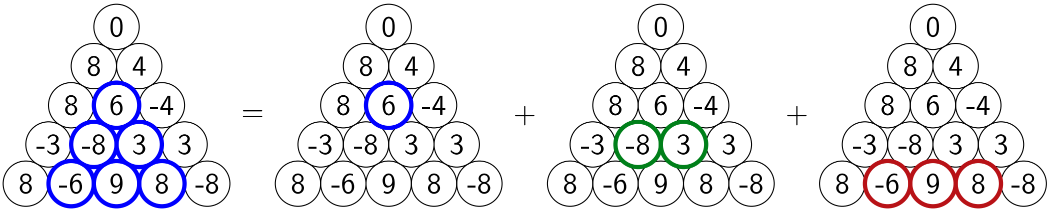
Naive attempt: for each package sum up the values in the pile below that package.

Running time: $\Omega(k^4)$.

(There are $\sim k^2/4$ packages in the top half and each of them has $\sim k^2/4$ packages below them.)

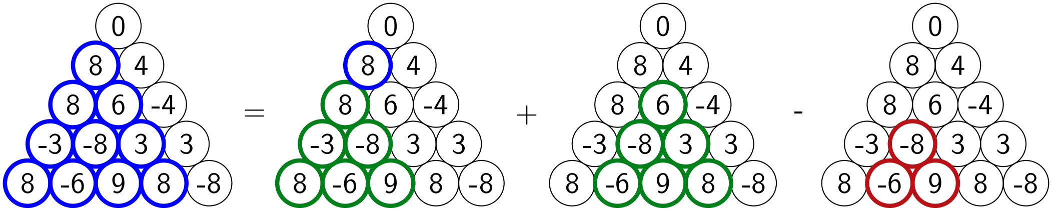
Alice's Accumulation – partial sums

Idea: precompute partial sums in each row (in time $O(k^2)$). Then we get a solution with running time $O(k^3)$:



Alice's Accumulation – recursion

Idea: find a recursion.



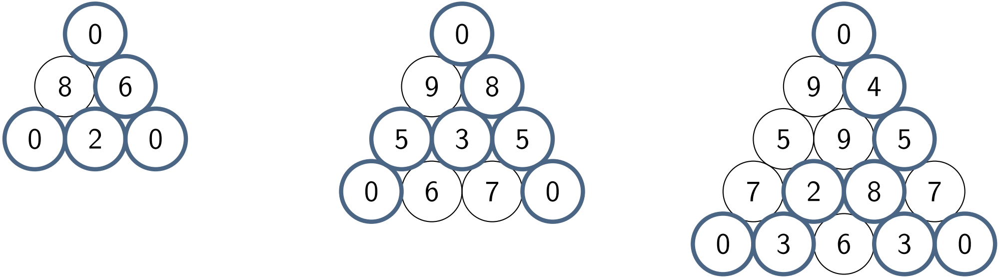
Alice's Accumulation – dynamic program

Hence, dynamic programming. Running time: $O(k^2)$.

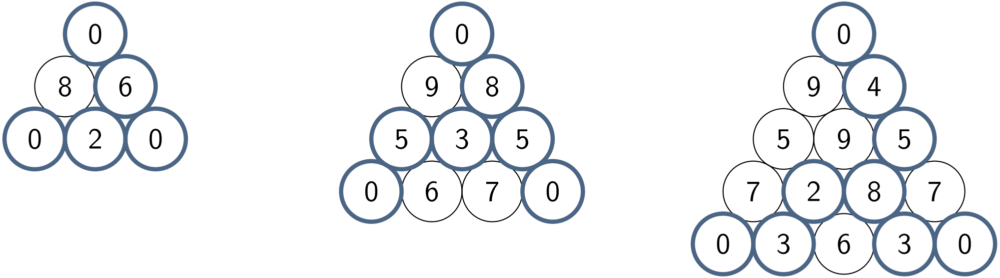
```
// Given: D[i][j] = value of the j-th ball in the i-th row
// Compute: P[i][j] = sum of sub-pile rooted at (i,j)

for (int i = k-1; i >= 0; --i) {
  for (int j = 0; j < i; ++j) {
    if (i == k-1)
      P[i][j] = D[i][j];
    else if (i == k-2)
      P[i][j] = D[i][j] + P[i+1][j] + P[i+1][j+1];
    else
      P[i][j] = D[i][j] + P[i+1][j] + P[i+1][j+1] - P[i+2][j+1];
  }
}
```

Bob's Burden – Understanding the problem

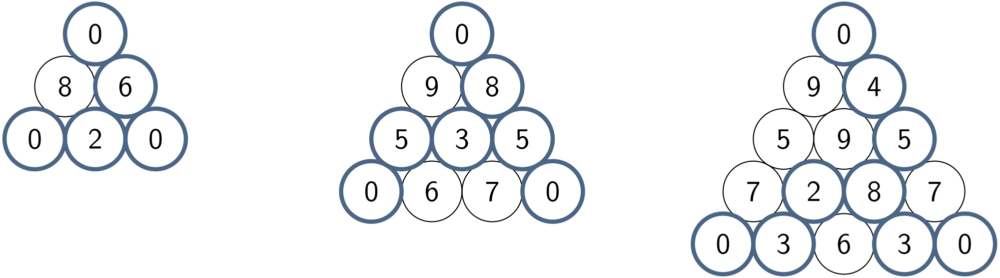


Bob's Burden – Understanding the problem



Model: graph on B_{ij}, edge $\iff$ disks touch

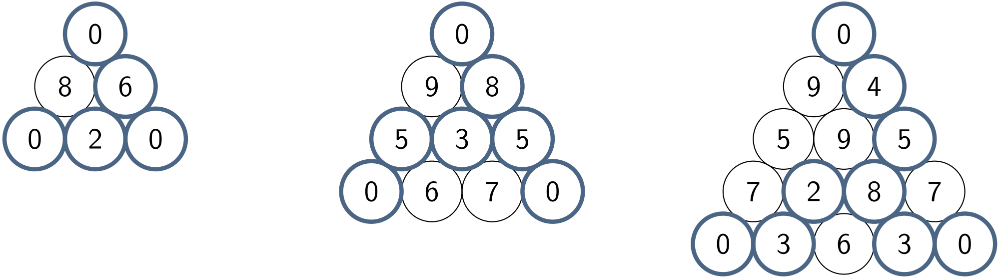
Bob's Burden – Understanding the problem



Model: graph on B_{ij}, edge $\iff$ disks touch

$\implies$ looking for a tree spanning the apices (not a spanning tree of the whole graph)

Bob's Burden – Understanding the problem



Model: graph on B_{ij}, edge $\iff$ disks touch

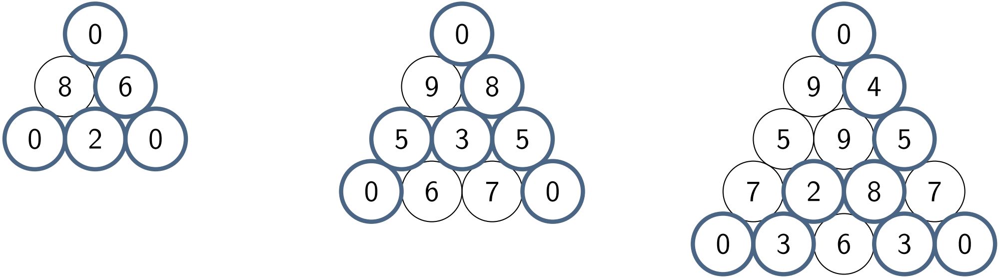
$\implies$ looking for a tree spanning the apices (not a spanning tree of the whole graph)

Q1: How does such a tree look like?

Q2: How to model weights?

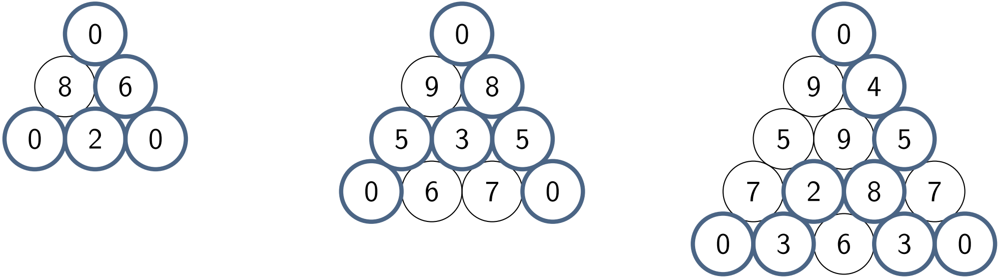
Q3: How to compute an optimum tree?

Bob's Burden – How does the tree look like?



Obs. The tree consists of a center vertex c and optimum paths between c and the three apices.

Bob's Burden – How does the tree look like?

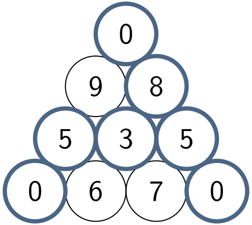


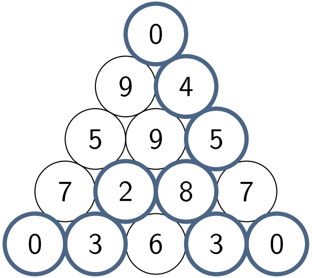
Obs. The tree consists of a center vertex c and optimum paths between c and the three apices.

Remark. This property is crucial for an efficient solution. The general Minimum Steiner Tree problem is NP-hard.

Bob's Burden – Modeling Weights and Distances

Task: Find center ball
with minimum
weight + distances.

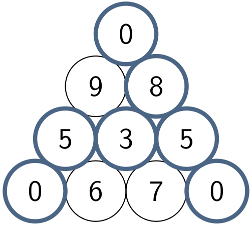


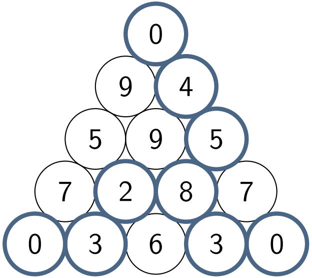


Bob's Burden – Modeling Weights and Distances

Task: Find center ball
with minimum
weight + distances.

Weight: own weight of ball

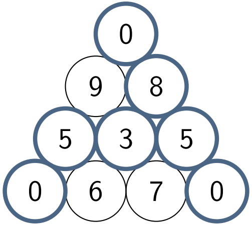


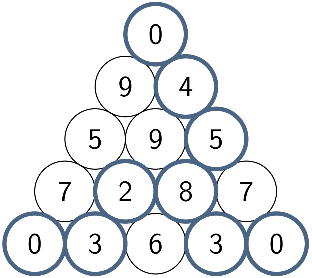


Bob's Burden – Modeling Weights and Distances

Task: Find center ball with minimum *weight + distances.*

Weight: own weight of ball
Distances: to triangle apex

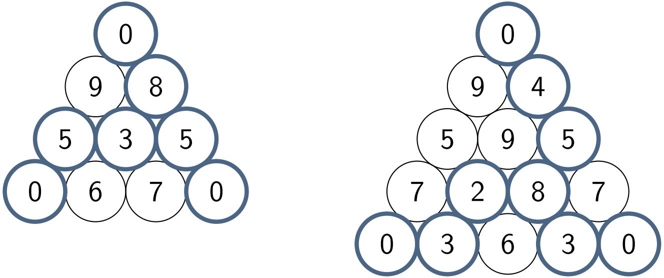




Bob's Burden – Modeling Weights and Distances

Task: Find center ball
with minimum
weight + distances.

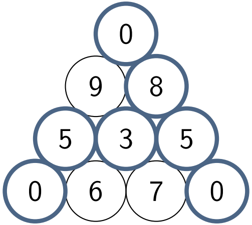
Weight: own weight of ball
Distances: to triangle apex
Center: may not be unique

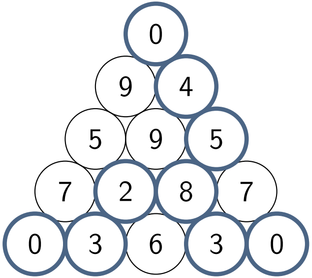


Bob's Burden – Modeling Weights and Distances

Task: Find center ball
with minimum
weight + distances.

Weight: own weight of ball
Distances: to triangle apex
Center: may not be unique





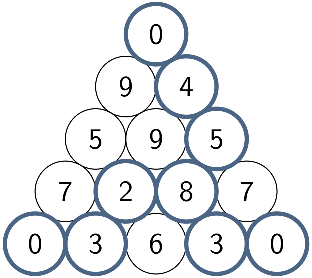
Graph model:

- B_{in} and B_{out} for each ball B,
- interior edge with the ball's weight,
- incoming/outgoing 0-edges to neighbors.

Bob's Burden – Computation 1st Subtask

First subtask: $k \leq 40 \;\Rightarrow\; \approx 800$ balls

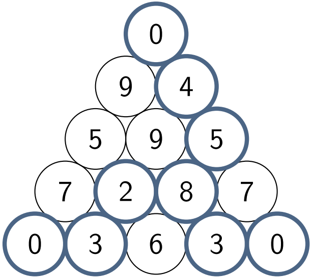
For each candidate ball B,
we can compute the distances to each triangle apex:



Bob's Burden – Computation 1st Subtask

First subtask: $k \leq 40 \;\Rightarrow\; \approx 800$ balls

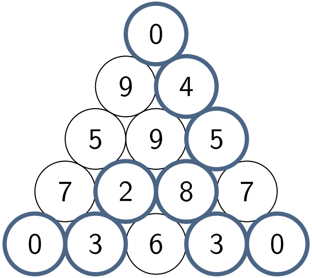
For each candidate ball B,
we can compute the distances to each triangle apex:
Either by running Dijkstra from B_{out}.
In this case we read the distances stored at
$B11_{\text{out}}, Bk1_{\text{out}}, Bkk_{\text{out}}$.



Bob's Burden – Computation 1st Subtask

First subtask: $k \leq 40 \;\Rightarrow\; \approx 800$ balls

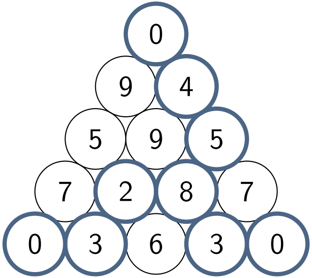
For each candidate ball B,
we can compute the distances to each triangle apex:
Either by running Dijkstra from B_{out}.
In this case we read the distances stored at $B11_{\mathrm{out}}, Bk1_{\mathrm{out}}, Bkk_{\mathrm{out}}$.
Or by running a MinCost MaxFlow (SSP version) from B_{out} to a sink reachable from $B11_{\mathrm{out}}, Bk1_{\mathrm{out}}, Bkk_{\mathrm{out}}$.
In this case we get the sum of the distances with `find_flow_cost(G)`.



Bob's Burden – Full Solution

Full solution: $k \leq 800 \;\Rightarrow\; \approx 320'000$ balls

We probably still have to look at each ball as a candidate.
How can we avoid the many (costly) Dijkstra runs?

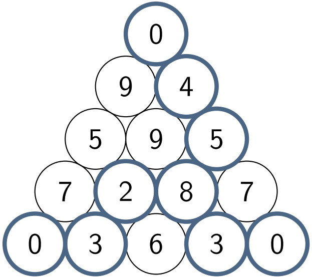


Bob's Burden – Full Solution

Full solution: $k \leq 800 \;\Rightarrow\; \approx 320'000$ balls

We probably still have to look at each ball as a candidate.
How can we avoid the many (costly) Dijkstra runs?

Do Dijkstra **only 3 times**, from $B11_{\mathrm{out}}$, $Bk1_{\mathrm{out}}$ and Bkk_{out}!

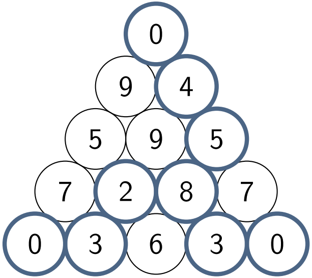


Bob's Burden – Full Solution

Full solution: $k \leq 800 \;\Rightarrow\; \approx 320'000$ balls

We probably still have to look at each ball as a candidate.
How can we avoid the many (costly) Dijkstra runs?

Do Dijkstra **only 3 times**, from $B11_{\mathrm{out}}$, $Bk1_{\mathrm{out}}$ and Bkk_{out}!
Details: Use 3 exterior property maps `distmap`i, $i = 1, 2, 3$.
Access distances with `distmap`i`[`B_{in}`]`.
Add the weight of the ball B.

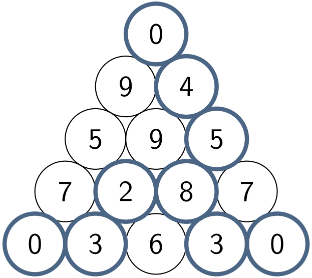


Bob's Burden – Full Solution

Full solution: $k \leq 800 \;\Rightarrow\; \approx 320'000$ balls

We probably still have to look at each ball as a candidate.
How can we avoid the many (costly) Dijkstra runs?

Do Dijkstra **only 3 times**, from $B11_{\text{out}}$, $Bk1_{\text{out}}$ and Bkk_{out}!
Details: Use 3 exterior property maps `distmap`i, $i = 1, 2, 3$.
Access distances with `distmap`i`[`B_{in}`]`.
Add the weight of the ball B.



Remark: Simple testdata available on the judge.

Carol’s Configuration – boundary only

This case is quite easy to solve “by hand”.

Carol’s Configuration – boundary only

This case is quite easy to solve “by hand”.

Note that the balls on the boundary form a cyclic structure.

Carol's Configuration – boundary only

This case is quite easy to solve "by hand".

Note that the balls on the boundary form a cyclic structure.

If k is even, this cycle has odd length:

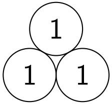
Carol's Configuration – boundary only

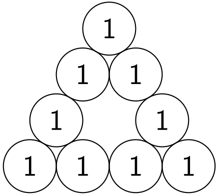
This case is quite easy to solve "by hand".

Note that the balls on the boundary form a cyclic structure.

If k is even, this cycle has odd length:

The cycle stays connected
if and only if all radii are 1.





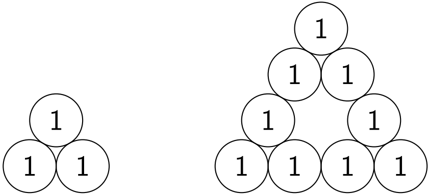
Carol’s Configuration – boundary only

This case is quite easy to solve “by hand”.

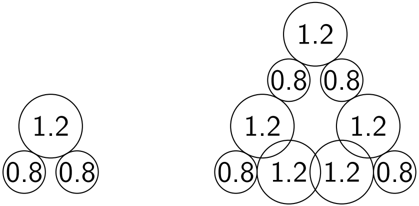
Note that the balls on the boundary form a cyclic structure.

If k is even, this cycle has odd length:

The cycle stays connected
if and only if all radii are 1.



If any radius is different from 1,
then it will be disconnected or intersecting.



Carol's Configuration – boundary only

If k is odd, the cycle has even length:

Carol's Configuration – boundary only

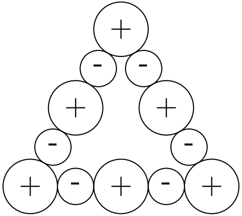
If k is odd, the cycle has even length:

- Set the radius of B_{11} (and the other corners) to $1 + x$, $0 \leq x \leq 1$. It cannot be smaller because the two neighbors would intersect otherwise.
- The radii on the cycle alternate between $1 + x$ and $1 - x$.

Carol's Configuration – boundary only

If k is odd, the cycle has even length:

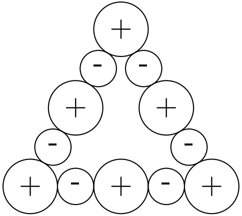
- Set the radius of B_{11} (and the other corners) to $1 + x$, $0 \leq x \leq 1$. It cannot be smaller because the two neighbors would intersect otherwise.
- The radii on the cycle alternate between $1 + x$ and $1 - x$.



Carol's Configuration – boundary only

If k is odd, the cycle has even length:

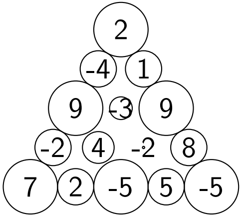
- Set the radius of B_{11} (and the other corners) to $1+x$, $0 \leq x \leq 1$. It cannot be smaller because the two neighbors would intersect otherwise.
- The radii on the cycle alternate between $1+x$ and $1-x$.



The objective is a linear function of the form $(1+x)c_1 + (1-x)c_2$.
Any monotone function on a closed interval I takes its maximum value on the boundary of I, i.e., at $x=0$ or $x=1$.

Carol's Configuration – full solution

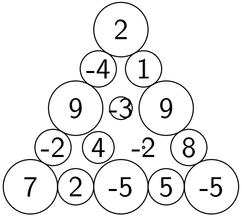
For the general case we will formulate a Linear Program.



Carol's Configuration – full solution

For the general case we will formulate a Linear Program.

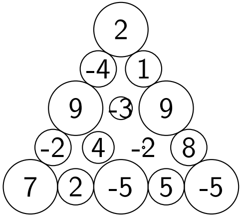
$$
\begin{array}{lllll}
\text{maximize} & \sum r_{ij} v_{ij} & & \\
\text{subject to} & r_{ij} + r_{i'j'} & \leq 2, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_1(ij) \\
& r_{ij} + r_{i'j'} & \leq 2\sqrt{3}, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_2(ij) \\
& r_{i1} + r_{(i+1)1} & = 2 & \text{for all } i < k \\
& r_{ii} + r_{(i+1)(i+1)} & = 2 & \text{for all } i < k \\
& r_{kj} + r_{k(j+1)} & = 2 & \text{for all } j < k
\end{array}
$$



Carol's Configuration – full solution

For the general case we will formulate a Linear Program.

$$
\begin{array}{lll}
\text{maximize} & \sum r_{ij} v_{ij} & \\
\text{subject to} & r_{ij} + r_{i'j'} \leq 2, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_1(ij) \\
& r_{ij} + r_{i'j'} \leq 2\sqrt{3}, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_2(ij) \\
& r_{i1} + r_{(i+1)1} = 2 & \text{for all } i < k \\
& r_{ii} + r_{(i+1)(i+1)} = 2 & \text{for all } i < k \\
& r_{kj} + r_{k(j+1)} = 2 & \text{for all } j < k
\end{array}
$$

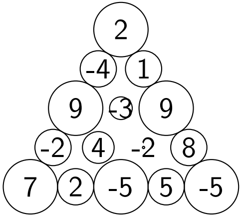


$N_1(ij)$: The set of direct neighbors of B_{ij}.
$N_2(ij)$: The set of indirect neighbors of B_{ij}.
(All the balls that might intersect with B_{ij}.)

Carol's Configuration – full solution

For the general case we will formulate a Linear Program.

$$
\begin{array}{lll}
\text{maximize} & \sum r_{ij} v_{ij} & \\
\text{subject to} & r_{ij} + r_{i'j'} \leq 2, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_1(ij) \\
 & r_{ij} + r_{i'j'} \leq 2\sqrt{3}, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_2(ij) \\
 & r_{i1} + r_{(i+1)1} = 2 & \text{for all } i < k \\
 & r_{ii} + r_{(i+1)(i+1)} = 2 & \text{for all } i < k \\
 & r_{kj} + r_{k(j+1)} = 2 & \text{for all } j < k
\end{array}
$$



$N_1(ij)$: The set of direct neighbors of B_{ij}.
$N_2(ij)$: The set of indirect neighbors of B_{ij}.
(All the balls that might intersect with B_{ij}.)

For all ij we have $|N_1(ij)| \leq 6$ and $|N_2(ij)| \leq 6$.