Application of LP to Network Design - Project 1

Harshavardhan Nalajala

June 17 2017

1 Description

Purpose of the project is to create software that estimates the optimal cost of designing a network using linear programming. We need to identify input, objective function, and constraints in order to create the network with capacity links.

1.1 Input

Number of nodes given is 30. Cost of using a link from $\mathrm{node_i}$ to $\mathrm{node_j}$ is given by $\mathrm{a_{ij}}$. Source rate is given by $\mathrm{b_{ij}}$ from a $\mathrm{node_i}$ to $\mathrm{node_j}$. Cost($\mathrm{a_{ij}}$) and source rate($\mathrm{b_{ij}}$) are obtained using the rules described.

1.2 Objective Function

Objective of the design is to minimize the cost of designing a network which satisfies the capacity requirements. Objective function is given by

$$Z_{OPT} = \sum_{k,l} b_{k,l}(a_{k,i} + a_{i,l}) \forall i \in k, l \tag{1}$$

1.3 Constraints

$$a_{ij} \geq 0 \tag{2}$$

$$b_{ij} \geq 0 \tag{3}$$

2 Algorithm

Algorithm to design the network takes matrix of costs($\mathrm{a_{ij}}$), matrix of source rate($\mathrm{b_{ij}}$) as inputs and generates the graph given by capacity matrix ($\mathrm{z_{ij}}$). Description of the algorithm is given below.

- Create the shortest path connectivity matrix of the graph using Bellman ford algorithm and store it in a_{ij}
- Calculate the capacity matrix using $z_{ij} = b_{ij} * a_{ij}$

3 Output

The software package is run using 30 nodes, with K ranging from 3 to 15. Following is part of the output. Full output can be found in the ReadMe section.

k(3),	optz(6726)	density(0.889655)
k(4),	optz(5779)	density(0.889655)
k(5),	optz(5000)	density(0.889655)
k(6),	optz(4855)	density(0.889655)
k(7),	optz(4497)	density(0.889655)
k(8),	optz(4504)	density(0.889655)
k(9),	optz(4197)	density(0.889655)
k(10),	optz(3992)	density(0.889655)
k(11),	optz(3881)	density(0.889655)
k(12),	optz(3809)	density(0.889655)
k(13),	optz(3769)	density(0.889655)
k(14),	optz(3683)	density(0.889655)
k(15),	optz(3704)	density(0.889655)

4 Analysis

As value k increases, number of cost of links reduce and hence total cost of the network reduces. Density of the network is a constant since the number of zero capacity nodes seem to be constant and not changing with k.

5 ReadMe

Run Makefile using following command in the directory.

```
make -f Makefile
```

Makefile generates the executabla(lp). Run lp in command window. Input number of nodes. Output will be printed on the screen. Output can be redirected to a file.

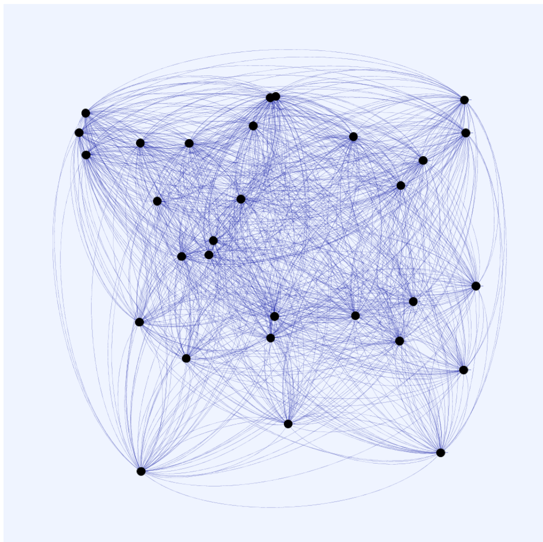

Figure 1: Graph obtained with k = 3

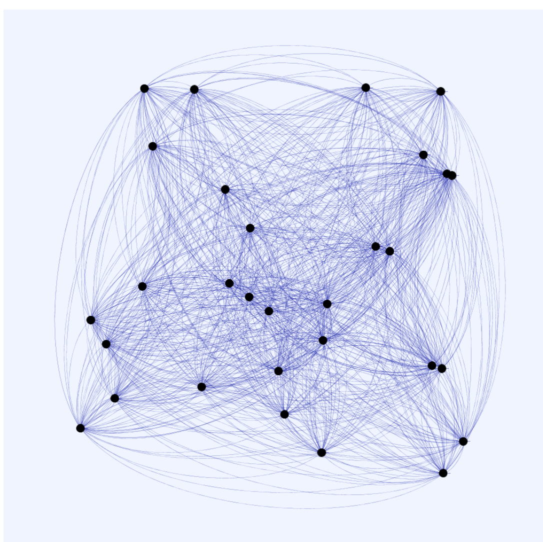

Figure 2: Graph obtained with k = 9

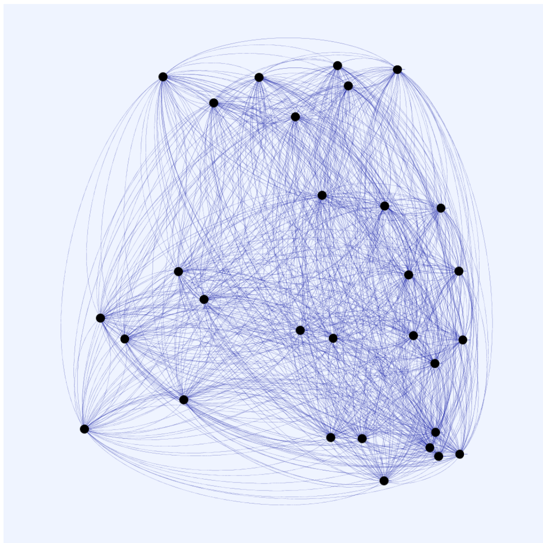

Figure 3: Graph obtained with k = 15

6 Appendix

File: lp.h

```
#ifndef _LP_
#define _LP_

#define K_START 3
#define K_END 16

#ifndef DEBUG
        #define printf(p, ...)
#else
#endif
#endif
```

File: LP.c

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include "lp.h"
```

```
void print(int nodes, int a[][nodes]) {
        int i = 0;
        int j = 0;
        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        printf("%d ", a[i][j]);
                }
                printf("\n");
        }
}

void initializeA(int nodes, int a[][nodes], int k) {
        int i = 0;
        int j = 0;
        int l = 0;

        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        a[i][j] = INT_MAX;
                }
        }

        // for all i
        //      get k random l
        //              set a[i][l] = 1
        //      if j is not in l, a[i][j] = 300
        for (i = 0; i < nodes; i++) {
                a[i][i] = 0;
                for (l = 0; l < k; l++) {
                        int j_i = rand()%nodes;
                        while (j_i == i) {
                                j_i = rand()%nodes;
                        }
                        a[i][j_i] = 1;
                }
                for (j = 0; j < nodes; j++) {
                        if (a[i][j] != 1 && i != j) {
                                a[i][j] = 300;
                        }
                }
```

```
        }
}

void initializeB(int nodes, int b[][nodes], char *id) {
        int i = 0;
        int j = 0;

        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        b[i][j] = 0;
                        b[i][j] = abs((id[i]-'0') - (id[j]-'0'));
                }
        }
}

void get_capacities(int nodes, int z[][nodes],
                int a[][nodes], int b[][nodes]) {
        int i = 0;
        int j = 0;
        int l = 0;
        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        if (i == j) {
                                z[i][j] = 0;
                        } else {
                                z[i][j] = b[i][j]*a[i][j];
                        }
                }
        }

        int m = 0;
        while (m < nodes) {
        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        if (i != j) {
                                int temp = INT_MAX;
                                for (l = 0; l < nodes; l++) {
                                        temp = b[i][j]*(a[i][l] + a[l][j])
                                        if (temp < z[i][j]) {
                                                z[i][j] = temp;
```

```
                                        }
                                }
                        }
                }
        }
        m++;
        }
}

int get_opt_z(int nodes, int z[][nodes]) {
        int i = 0;
        int j = 0;
        int z_opt = 0;

        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        z_opt += z[i][j];
                }
        }

        return z_opt;
}

void form_topologies(int nodes, int a[][nodes]) {
        int i = 0;
        int j = 0;
        int k = 0;
        int l = 0;

        while (l < nodes) {
                for (i = 0; i < nodes; i++) {
                        if (i != j) {
                                for (j = 0; j < nodes; j++) {
                                        int temp = INT_MAX;
                                        for (k = 0; k < nodes; k++) {
                                                temp = a[i][k] + a[k][j];
                                                if (temp < a[i][j]) {
                                                        a[i][j] = temp;
                                                }
                                        }
```

```
                                }
                        }
                }
                l++;
        }
}

float get_density(int nodes, int z[][nodes]) {
        int i = 0;
        int j = 0;
        float n = 0;
        float total = (float)(nodes * (nodes-1));
        float density;

        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        if (z[i][j] != 0) {
                                n++;
                        }
                }
        }

        //printf("Number of non zero capacities: %f\n", n);
        density = n/total;
        return density;
}

void printGraph(int nodes, int z[][nodes]) {
        int i = 0;
        int j = 0;
        int k = 0;
        for (i = 0; i < nodes; i++) {
                for (j = 0; j < nodes; j++) {
                        printf("%d      %d      Directed
%d      %d\n", i, j, k++, z[i][j]);
                }
        }
}

int main() {
```

```
        int nodes = 0;
        int k = K_START;
        int k_max = K_END;
        char *id = "2021346835202134683520213468 35";
        int index = k;

        scanf("%d", &nodes);
        for (index = k; index < k_max; index++) {
                int a[nodes][nodes];
                int b[nodes][nodes];

                initializeA(nodes, a, index);
                //printf("\nObtained Graph:\n");
                //print(nodes, a);

                initializeB(nodes, b, id);
                //printf("\nObtained B:\n");
                //print(nodes, b);
                form_topologies(nodes, a);
                printf("\nShortest Path Graph:\n");
                print(nodes, a);

                int z[nodes][nodes];
                get_capacities(nodes, z, a, b);
                printf("\nObtained Capacity Matrix:\n");
                print(nodes, z);

                int z_opt = get_opt_z(nodes, z);
                float density = get_density(nodes, z);
                printf("\nFor k(%d), z_opt(%d) density(%f)\n", index, z_opt
        }
        return 1;
}
```

File: Makefile

```
lp: lp.o
        gcc -o lp lp.o
lp.o: LP.c lp.h
        gcc -c -Wall -DDEBUG LP.c -I.
```

```
clean:
	rm -rf lp.o lp
```

6.1 Full Output with nodes 30

```
Shortest Path Graph:
0 2 3 4 3 3 4 1 3 2 3 4 4 2 4 3 5 2 3 1 4 3 5 1 4 3 2 4 2 3
3 0 1 2 4 3 4 2 1 2 1 5 5 3 4 3 5 3 3 3 4 4 5 2 2 3 2 5 3 3
3 2 0 2 3 3 3 3 1 2 2 4 4 2 4 2 5 3 3 3 4 3 4 1 1 2 3 4 2 4
1 3 3 0 4 1 5 2 4 3 4 4 5 2 4 3 6 3 4 1 5 3 6 2 4 4 2 5 3 3
3 2 1 3 0 4 4 4 2 2 3 5 1 3 5 3 5 4 3 2 4 4 5 2 2 3 3 1 3 4
2 2 3 4 3 0 4 1 3 2 3 4 4 2 4 3 5 2 3 1 4 3 5 1 4 3 2 4 2 3
2 3 2 1 3 2 0 2 3 3 1 5 4 3 4 3 4 3 2 2 3 4 1 3 3 4 2 2 3 3
2 3 3 4 2 3 3 0 4 1 4 4 3 2 4 3 4 1 2 1 3 3 4 2 3 2 2 3 2 3
2 4 2 1 3 2 3 2 0 1 1 5 4 3 4 3 4 3 2 2 3 4 4 3 3 2 2 4 3 3
4 3 3 3 2 2 2 3 3 0 3 4 3 3 3 2 3 3 1 3 2 3 3 2 2 1 1 3 2 2
3 3 1 3 3 2 4 1 2 2 0 4 4 3 3 2 5 2 3 2 4 3 5 2 2 3 1 4 3 2
2 4 4 1 3 2 4 1 5 2 5 0 4 3 5 4 2 2 3 2 1 4 2 3 3 3 3 3 3 4
2 4 3 4 3 3 3 3 4 1 4 4 0 2 4 3 4 4 2 1 3 3 4 3 3 2 2 4 3 3
3 3 1 3 3 3 3 1 2 2 3 2 4 0 2 3 4 2 3 2 3 1 4 2 2 3 3 4 3 4
3 3 2 2 4 1 1 2 3 3 2 3 5 1 0 4 5 3 3 2 4 2 2 2 3 4 3 3 3 4
3 1 2 2 4 1 4 2 1 2 2 5 5 3 5 0 5 3 3 2 4 4 5 2 3 3 3 5 3 4
3 3 3 2 3 3 3 1 3 2 4 1 4 2 4 2 0 2 3 2 2 3 3 3 1 2 3 4 3 4
4 2 2 4 1 2 5 3 3 3 3 4 2 2 4 2 6 0 4 3 5 3 6 1 3 4 4 2 1 5
4 3 3 4 3 2 3 3 3 2 4 3 4 2 4 2 2 2 0 3 1 3 2 3 1 2 3 3 1 4
1 3 2 4 4 2 4 2 3 3 4 3 5 1 3 2 5 3 3 0 4 2 5 2 3 4 1 5 3 2
4 3 4 3 3 3 3 2 4 1 4 2 4 3 4 3 1 3 2 3 0 4 1 3 2 2 2 2 3 3
3 4 2 2 4 2 2 2 3 3 3 1 5 1 1 4 3 3 4 3 2 0 3 3 3 4 4 4 4 5
5 2 3 4 2 3 4 4 3 2 3 4 3 3 5 3 3 3 1 4 2 4 0 4 2 3 3 1 2 4
4 1 2 3 3 2 4 2 2 3 2 3 4 1 3 2 5 2 4 3 4 2 5 0 3 4 3 4 1 4
4 2 2 3 2 2 2 2 2 3 3 3 3 1 3 1 5 3 4 3 4 2 3 2 0 1 4 3 3 5
3 2 2 2 1 3 1 3 3 3 2 4 2 2 4 3 5 3 3 3 4 3 2 1 3 0 3 2 2 4
3 2 3 3 4 1 3 2 2 3 3 3 5 3 2 1 4 3 2 2 3 2 4 2 3 4 0 5 3 1
4 1 2 3 1 3 3 3 2 1 2 5 2 4 4 3 4 4 2 3 3 4 4 3 3 2 2 0 3 3
3 2 3 3 2 1 5 2 2 3 3 5 3 3 5 1 6 1 4 2 5 4 6 2 4 4 3 3 0 4
4 4 3 3 4 2 2 3 4 3 3 2 5 2 1 3 3 3 1 3 2 1 3 3 2 3 4 4 2 0

Obtained Capacity Matrix:
0 4 0 4 3 6 16 6 3 6 0 8 0 2 4 6 20 12 3 3 0 6 0 1 4 6 8 24 2 9
```

```
6 0 2 2 12 12 24 16 3 10 2 0 10 3 12 12 30 24 9 15 8 0 10 2 6 12 12 40 9 15
0 4 0 2 3 6 12 18 1 6 0 8 0 2 4 4 20 18 3 9 0 6 0 1 1 4 12 24 2 12
1 3 3 0 8 3 25 14 8 12 4 4 5 0 8 9 30 21 8 4 5 3 6 0 8 12 10 35 6 12
3 6 1 6 0 4 12 20 0 4 3 15 1 6 0 3 15 20 0 4 4 12 5 4 0 3 9 5 0 8
4 8 6 12 3 0 8 4 3 2 6 16 8 6 4 0 10 8 3 1 8 12 10 3 4 0 4 16 2 3
8 18 8 5 9 4 0 4 9 3 4 30 16 15 12 6 0 6 6 2 12 24 4 15 9 8 0 4 9 3
12 24 18 28 10 12 6 0 20 3 24 32 18 14 20 12 8 0 10 3 18 24 24 14 15 8 4 0
2 12 2 2 0 2 9 10 0 2 1 15 4 6 0 3 12 15 0 4 3 12 4 6 0 2 6 20 0 6
12 15 9 12 4 2 2 9 6 0 9 20 9 12 6 2 3 9 2 0 6 15 9 8 4 1 1 9 4 0
0 6 0 3 3 4 16 6 2 6 0 8 0 3 3 4 20 12 3 6 0 6 0 2 2 6 4 24 3 6
4 0 8 1 9 8 24 8 15 10 10 0 8 3 15 16 12 16 9 10 2 0 4 3 9 12 18 24 9 20
0 8 0 4 3 6 12 18 4 3 0 8 0 2 4 6 16 24 2 3 0 6 0 3 3 4 8 24 3 9
3 3 1 0 6 9 15 7 4 8 3 2 4 0 4 9 20 14 6 8 3 1 4 0 4 9 15 28 6 16
3 9 2 4 0 1 3 10 0 6 2 9 5 2 0 4 15 15 0 4 4 6 2 4 0 4 9 15 0 8
6 4 4 6 4 0 8 8 1 2 4 20 10 9 5 0 10 12 3 2 8 16 10 6 3 0 6 20 3 4
12 18 12 10 9 6 0 2 9 2 16 6 16 10 12 4 0 4 9 2 8 18 12 15 3 4 0 8 9 4
24 16 12 28 5 8 10 0 15 9 18 32 12 14 20 8 12 0 20 9 30 24 36 7 15 16 8 0 5
4 9 3 8 0 2 9 15 0 4 4 9 4 4 0 2 6 10 0 6 1 9 2 6 0 2 9 15 0 8
3 15 6 16 8 2 4 6 6 0 12 15 15 4 6 2 5 9 6 0 12 10 15 8 6 4 1 15 6 0
0 6 0 3 3 6 12 12 4 3 0 4 0 3 4 6 4 18 2 9 0 8 0 3 2 4 8 12 3 9
6 0 4 2 12 8 12 16 9 15 6 0 10 1 3 16 18 24 12 15 4 0 6 3 9 16 24 32 12 25
0 4 0 4 2 6 16 24 3 6 0 8 0 3 5 6 12 18 1 12 0 8 0 4 2 6 12 6 2 12
4 1 2 0 6 6 20 14 4 12 2 3 4 0 6 6 25 14 8 12 4 2 5 0 6 12 15 28 2 16
4 6 2 6 0 2 6 10 0 6 3 9 3 2 0 1 15 15 0 6 4 6 3 4 0 1 12 15 0 10
6 8 4 6 1 0 2 12 3 3 4 16 4 6 4 0 10 12 3 3 8 12 4 3 3 0 6 8 2 4
12 12 12 15 12 2 0 4 6 3 12 18 20 15 6 2 0 6 6 2 12 12 16 10 9 8 0 10 9 1
24 8 12 21 5 12 6 0 10 3 12 40 12 28 20 12 8 0 10 9 18 32 24 21 15 8 4 0 15
3 6 3 6 0 1 15 10 0 6 3 15 3 6 0 1 18 5 0 4 5 12 6 4 0 4 9 15 0 8
12 20 9 12 8 2 2 9 8 0 9 10 15 8 2 3 3 9 2 0 6 5 9 12 4 3 4 12 4 0

For k(3), z_opt(6726) density(0.889655)

Shortest Path Graph:
0 4 2 3 2 1 3 3 3 2 2 2 3 2 2 2 2 2 1 3 3 3 1 1 3 2 3 3 3 2
4 0 4 2 3 4 2 3 2 2 3 3 1 3 3 2 1 3 3 3 2 2 3 3 2 2 1 1 4 5
3 4 0 4 3 3 4 2 1 3 2 3 2 2 2 2 3 4 2 3 3 1 3 2 1 2 2 1 3 4
3 5 3 0 3 4 4 3 1 3 2 1 3 3 3 2 3 1 2 3 2 2 2 2 3 2 2 3 4 5
2 3 2 4 0 3 4 1 2 3 3 3 2 4 2 2 3 4 3 3 4 3 3 3 1 2 2 1 2 3
3 4 3 2 3 0 4 2 3 2 3 1 3 3 3 1 3 3 3 2 2 2 2 2 2 3 3 3 3 1
3 5 2 4 2 3 0 3 1 2 2 3 3 2 3 2 3 2 2 1 3 3 1 2 2 2 2 3 4 4
```

```
1 2 1 3 3 2 4 0 2 3 3 3 3 3 2 2 3 3 2 3 3 2 2 2 1 2 3 2 1 2
2 4 3 3 2 3 3 2 0 3 1 2 4 2 2 2 2 3 1 3 3 2 2 1 3 1 4 3 3 4
4 4 2 2 3 4 2 3 2 0 3 2 1 3 2 2 1 2 3 3 2 2 1 3 3 1 3 2 3 4
2 3 2 4 4 3 4 1 3 2 0 3 3 2 3 2 3 2 3 3 3 1 1 3 2 2 3 2 2 3
3 4 2 4 2 3 4 2 3 2 2 0 3 2 3 1 3 2 2 2 1 1 1 2 2 2 3 2 3 4
3 4 3 2 3 3 2 2 1 3 2 3 0 2 3 2 1 3 2 3 2 1 3 2 3 2 2 1 3 4
3 5 4 3 2 1 4 3 3 2 2 2 3 0 3 2 3 4 2 1 1 3 3 3 2 3 2 3 4 2
3 2 1 2 3 3 2 2 2 2 3 3 3 3 0 1 1 3 3 2 2 2 3 2 2 2 3 2 1 2
4 4 3 3 2 3 3 3 2 1 2 3 2 2 2 0 2 3 3 1 3 3 2 1 1 2 2 3 3 4
3 4 3 1 2 4 1 3 2 2 2 2 3 3 2 2 0 2 2 2 1 3 2 3 3 1 3 3 3 4
2 4 3 3 2 3 3 3 3 2 2 3 2 2 2 3 2 0 1 3 1 3 3 1 2 2 1 3 3 4
1 3 2 2 1 2 2 2 3 3 3 3 3 3 1 2 1 3 0 3 2 3 2 2 2 2 3 2 2 3
3 4 3 4 1 2 4 2 2 2 3 3 2 1 2 2 3 4 3 0 2 3 3 3 1 2 1 2 3 3
1 3 2 2 1 2 2 2 3 3 3 3 3 3 1 2 1 3 0 3 2 3 2 2 2 2 3 2 2 3
3 4 3 4 1 2 4 2 2 2 3 3 2 1 2 2 3 4 3 0 2 3 3 3 1 2 1 2 3 3
2 4 3 3 1 3 3 2 3 1 1 3 2 3 2 3 2 3 1 4 0 2 2 3 2 2 3 2 3 4
2 3 2 4 3 2 4 1 2 3 3 2 2 1 2 2 3 3 3 2 2 0 2 3 2 1 2 1 2 3
3 5 1 3 3 4 3 3 2 1 3 3 2 3 3 1 2 1 2 2 2 2 0 2 2 2 2 2 4 5
3 4 3 2 3 2 2 2 3 3 1 2 4 1 2 2 1 3 3 2 2 2 2 0 3 1 3 3 3 3
3 3 2 3 3 4 3 3 1 2 2 2 3 3 1 1 2 3 2 2 3 3 2 2 0 1 3 3 2 3
4 3 2 3 3 4 3 3 3 2 3 1 3 3 1 1 2 2 3 2 2 2 1 2 2 0 3 3 2 3
4 4 3 3 4 4 3 3 2 1 3 2 1 3 2 2 2 3 3 3 3 2 2 3 1 1 0 2 3 4
3 5 4 3 3 4 3 3 1 2 2 3 1 3 3 1 2 4 2 2 3 2 3 2 2 2 1 0 4 5
2 1 2 2 3 2 3 1 3 3 2 3 2 2 3 3 2 3 3 3 2 3 3 1 2 2 2 2 0 1
2 3 2 1 2 1 4 1 2 2 2 2 3 4 3 2 3 2 2 3 1 3 3 3 2 3 3 3 2 0
```

Obtained Capacity Matrix:

```
0 8 0 3 2 2 12 18 3 6 0 4 0 2 2 4 8 12 1 9 0 6 0 1 3 4 12 18 3 6
8 0 8 2 9 16 12 24 6 10 6 0 2 3 9 8 6 24 9 15 4 0 6 3 6 8 6 8 12 25
0 8 0 4 3 6 16 12 1 9 0 6 0 2 2 4 12 24 2 9 0 2 0 2 1 4 8 6 3 12
3 5 3 0 6 12 20 21 2 12 2 1 3 0 6 6 15 7 4 12 2 2 2 0 6 6 10 21 8 20
2 9 2 8 0 3 12 5 0 6 3 9 2 8 0 2 9 20 0 6 4 9 3 6 0 2 6 5 0 6
6 16 6 6 3 0 8 8 3 2 6 4 6 9 3 0 6 12 3 2 4 8 4 6 2 0 6 12 3 1
12 30 8 20 6 6 0 6 3 2 8 18 12 10 9 4 0 4 6 1 12 18 4 10 6 4 0 6 12 4
6 16 6 21 15 8 8 0 10 9 18 24 18 21 10 8 6 0 10 9 18 16 12 14 5 8 6 0 5 6
2 12 3 6 0 3 9 10 0 6 1 6 4 4 0 2 6 15 0 6 3 6 2 2 0 1 12 15 0 8
12 20 6 8 6 4 2 9 4 0 9 10 3 12 4 2 1 6 6 0 6 10 3 12 6 1 3 6 6 0
0 6 0 4 4 6 16 6 3 6 0 6 0 2 3 4 12 12 3 9 0 2 0 3 2 4 12 12 2 9
6 0 4 4 6 12 24 16 9 10 4 0 6 2 9 4 18 16 6 10 2 0 2 2 6 8 18 16 9 20
0 8 0 2 3 6 8 12 1 9 0 6 0 2 3 4 4 18 2 9 0 2 0 2 3 4 8 6 3 12
```

```
3 5 4 0 4 3 20 21 6 8 2 2 3 0 6 6 15 28 4 4 1 3 3 0 4 9 10 21 8 8
3 6 1 4 0 3 6 10 0 4 3 9 3 6 0 1 3 15 0 4 2 6 3 4 0 2 9 10 0 4
8 16 6 9 2 0 6 12 2 1 4 12 4 6 2 0 4 12 3 1 6 12 4 3 1 0 4 12 3 4
12 24 12 5 6 8 0 6 6 2 8 12 12 15 6 4 0 4 6 2 4 18 8 15 9 2 0 6 9 4
12 32 18 21 10 12 6 0 15 6 12 24 12 14 10 12 4 0 5 9 6 24 18 7 10 8 2 0 15
1 9 2 4 0 2 6 10 0 6 3 9 3 6 0 2 3 15 0 6 2 9 2 4 0 2 9 10 0 6
9 20 9 16 2 2 4 6 4 0 9 15 6 4 4 2 3 12 6 0 6 15 9 12 2 2 1 6 6 0
0 8 0 3 1 6 12 12 3 3 0 6 0 3 2 6 8 18 1 12 0 4 0 3 2 4 12 12 3 12
4 0 4 4 9 8 24 8 6 15 6 0 4 1 6 8 18 24 9 10 4 0 4 3 6 4 12 8 6 15
0 10 0 3 3 8 12 18 2 3 0 6 0 3 3 2 8 6 2 6 0 4 0 2 2 4 8 12 4 15
3 4 3 0 6 6 10 14 6 12 1 2 4 0 4 6 5 21 6 8 2 2 2 0 6 3 15 21 6 12
3 9 2 6 0 4 9 15 0 4 2 6 3 6 0 1 6 15 0 4 3 9 2 4 0 1 9 15 0 6
8 12 4 9 3 0 6 12 3 2 6 4 6 9 1 0 4 8 3 2 4 8 2 6 2 0 6 12 2 3
16 24 12 15 12 8 0 6 6 1 12 12 4 15 6 4 0 6 9 3 12 12 8 15 3 2 0 4 9 4
18 40 24 21 15 16 6 0 5 6 12 24 6 21 15 4 4 0 10 6 18 16 18 14 10 8 2 0 20
2 3 2 4 0 2 9 5 0 6 2 9 2 4 0 3 6 15 0 6 2 9 3 2 0 2 6 10 0 2
6 15 6 4 4 1 4 3 4 0 6 10 9 16 6 2 3 6 4 0 3 15 9 12 4 3 3 9 4 0


For k(4), z_opt(5779) density(0.889655)

Shortest Path Graph:
0 1 1 3 2 2 2 1 2 2 2 2 4 3 3 2 2 2 1 2 2 1 3 2 2 2 3 2 3 3
3 0 3 2 3 1 1 2 1 2 2 2 3 2 3 1 3 2 3 2 2 2 2 3 3 1 2 3 2 2
3 3 0 2 1 2 1 2 2 2 2 2 4 2 2 2 3 2 3 1 2 1 2 2 2 3 2 1 2 3
3 1 3 0 1 2 2 2 2 2 3 2 3 2 1 2 3 2 3 3 1 2 1 3 2 2 3 3 2 2
2 3 2 2 0 1 2 1 2 1 3 3 3 1 2 1 2 2 2 2 2 2 3 3 2 2 3 2 3 2
3 2 3 2 3 0 1 2 2 2 2 3 2 2 3 1 3 2 3 2 1 3 2 3 2 1 2 2 2 1
2 2 3 1 2 3 0 3 2 1 3 3 3 2 2 2 2 2 3 3 2 2 1 2 3 2 1 2 1 2
2 3 3 2 2 3 3 0 1 2 2 2 4 3 3 2 2 1 1 3 2 1 3 2 1 2 2 2 3 3
2 2 3 1 2 3 3 3 0 1 2 1 3 3 2 2 3 3 3 3 2 1 2 2 3 2 2 2 3 2
1 2 2 3 1 2 2 2 2 0 2 3 2 2 3 2 3 3 2 2 3 2 3 3 3 1 3 2 3 1
3 3 2 2 1 2 3 2 2 2 0 3 4 1 2 1 2 2 3 1 2 3 3 2 2 3 2 1 2 3
2 2 3 2 3 3 2 3 2 2 3 0 3 2 1 1 2 2 3 2 1 3 3 2 2 2 1 1 2 2
1 2 2 3 2 3 2 2 2 3 3 3 0 3 2 2 2 2 1 3 1 2 3 2 1 2 3 1 2 3
3 3 1 3 1 2 2 2 2 2 3 2 3 0 1 2 1 3 3 2 2 2 3 2 2 3 3 1 2 2
3 3 3 2 3 4 3 2 1 2 3 1 2 2 0 1 3 2 3 2 2 2 3 2 3 2 2 2 1 1
2 2 2 1 2 3 3 3 2 1 2 3 3 1 2 0 2 1 3 1 2 3 2 2 3 2 2 2 3 2
2 3 1 2 2 3 1 3 3 2 3 3 4 2 2 1 0 2 2 2 2 2 2 1 2 3 2 1 2 3
1 2 2 2 1 2 3 2 3 2 1 3 4 2 3 2 2 0 2 2 3 2 3 1 3 2 1 2 3 3
```

```
3 2 2 2 2 3 2 3 1 2 3 2 4 2 3 1 1 2 0 2 1 2 3 2 1 2 3 2 3 3
2 2 3 1 2 3 3 3 1 2 2 2 4 2 2 1 2 1 3 0 2 2 2 2 3 2 1 2 3 3
2 1 3 2 2 2 1 3 1 2 2 2 4 3 3 2 3 1 3 3 0 2 2 2 1 2 2 2 2 3
2 3 3 2 2 3 3 3 3 1 1 1 3 2 2 2 3 3 2 2 2 0 3 1 2 2 2 1 2 2
2 3 2 2 2 3 3 3 1 1 2 2 3 1 2 3 2 1 3 3 3 1 0 2 3 2 2 2 3 2
1 2 2 1 1 2 3 2 2 2 3 3 4 2 2 2 2 3 1 3 2 2 2 0 2 3 4 1 2 3
3 2 3 3 1 2 2 2 2 2 2 3 4 2 2 2 3 2 3 3 1 3 3 2 0 1 3 1 2 3
3 3 3 2 2 3 1 1 1 2 1 2 4 2 3 1 3 2 2 2 3 2 2 3 2 0 2 2 2 3
1 2 2 2 2 3 2 2 2 3 2 3 4 3 2 2 1 3 2 3 2 2 3 1 2 1 0 1 2 3
2 2 3 2 2 3 2 3 2 2 3 2 3 3 1 2 3 2 2 3 1 3 3 1 1 2 3 0 1 2
2 3 3 2 2 3 2 2 2 1 2 3 3 2 2 1 3 2 2 2 2 3 3 1 2 1 3 1 0 2
2 3 2 2 2 3 3 1 2 3 3 3 1 1 2 2 2 2 2 1 2 2 3 2 2 3 2 1 2 0
```

Obtained Capacity Matrix:

```
0 2 0 3 2 4 8 6 2 6 0 4 0 3 3 4 8 12 1 6 0 2 0 2 2 4 12 12 3 9
6 0 6 2 9 4 6 16 3 10 4 0 6 2 9 4 18 16 9 10 4 0 4 3 9 4 12 24 6 10
0 6 0 2 1 4 4 12 2 6 0 4 0 2 2 4 12 12 3 3 0 2 0 2 2 6 8 6 2 9
3 1 3 0 2 6 10 14 4 8 3 2 3 0 2 6 15 14 6 12 1 2 1 0 4 6 15 21 4 8
2 9 2 4 0 1 6 5 0 2 3 9 3 2 0 1 6 10 0 4 2 6 3 6 0 2 9 10 0 4
6 8 6 6 3 0 2 8 2 2 4 12 4 6 3 0 6 8 3 2 2 12 4 9 2 0 4 8 2 1
8 12 12 5 6 6 0 6 6 1 12 18 12 10 6 4 0 4 9 3 8 12 4 10 9 4 0 4 3 2
6 8 6 6 3 0 2 8 2 2 4 12 4 6 3 0 6 8 3 2 2 12 4 9 2 0 4 8 2 1
8 12 12 5 6 6 0 6 6 1 12 18 12 10 6 4 0 4 9 3 8 12 4 10 9 4 0 4 3 2
12 24 18 14 10 12 6 0 5 6 12 16 24 21 15 8 4 0 5 9 12 8 18 14 5 8 4 0 15 9
2 6 3 2 0 3 9 15 0 2 2 3 3 6 0 2 9 15 0 6 2 3 2 4 0 2 6 10 0 4
3 10 6 12 2 2 2 6 4 0 6 15 6 8 6 2 3 9 4 0 9 10 9 12 6 1 3 6 6 0
0 6 0 2 1 4 12 12 2 6 0 6 0 1 2 2 8 12 3 3 0 6 0 2 2 6 8 6 2 9
4 0 6 2 9 12 12 24 6 10 6 0 6 2 3 4 12 16 9 10 2 0 6 2 6 8 6 8 6 10
0 4 0 3 2 6 8 12 2 9 0 6 0 3 2 4 8 12 1 9 0 4 0 2 1 4 12 6 2 9
3 3 1 0 2 6 10 14 4 8 3 2 3 0 2 6 5 21 6 8 2 2 3 0 4 9 15 7 4 8
3 9 3 4 0 4 9 10 0 4 3 3 2 4 0 1 9 10 0 4 2 6 3 4 0 2 6 10 0 2
4 8 4 3 2 0 6 12 2 1 4 12 6 3 2 0 4 4 3 1 4 12 4 6 3 0 4 8 3 2
8 18 4 10 6 6 0 6 9 2 12 18 16 10 6 2 0 4 6 2 8 12 8 5 6 6 0 2 6 3
6 16 12 14 5 8 6 0 15 6 6 24 24 14 15 8 4 0 10 6 18 16 18 7 15 8 2 0 15 9
3 6 2 4 0 3 6 15 0 4 3 6 4 4 0 1 3 10 0 4 1 6 3 4 0 2 9 10 0 6
6 10 9 4 4 3 3 9 2 0 6 10 12 8 4 1 2 3 6 0 6 10 6 8 6 2 1 6 6 0
0 2 0 2 2 4 4 18 1 6 0 4 0 3 3 4 12 6 3 9 0 4 0 2 1 4 8 12 2 9
4 0 6 2 6 12 18 24 9 5 2 0 6 2 6 8 18 24 6 10 4 0 6 1 6 8 12 8 6 10
0 6 0 2 2 6 12 18 1 3 0 4 0 1 2 6 8 6 3 9 0 2 0 2 3 4 8 12 3 6
1 2 2 0 2 6 15 14 4 8 3 3 4 0 4 6 10 21 2 12 2 2 2 0 4 9 20 7 4 12
```

```
3 6 3 6 0 2 6 10 0 4 2 9 4 4 0 2 9 10 0 6 1 9 3 4 0 1 9 5 0 6
6 12 6 6 2 0 2 4 1 2 2 8 8 6 3 0 6 8 2 2 6 8 4 9 2 0 4 8 2 3
4 12 8 10 6 6 0 4 6 3 8 18 16 15 6 4 0 6 6 3 8 12 12 5 6 2 0 2 6 3
12 16 18 14 10 12 4 0 10 6 18 16 18 21 5 8 6 0 10 9 6 24 18 7 5 8 6 0 5 6
2 9 3 4 0 3 6 10 0 2 2 9 3 4 0 1 9 10 0 4 2 9 3 2 0 1 9 5 0 4
6 15 6 8 4 3 3 3 4 0 9 15 3 4 4 2 2 6 4 0 6 10 9 8 4 3 2 3 4 0

For k(5), z_opt(5000) density(0.889655)

Shortest Path Graph:
0 3 2 1 2 2 2 3 2 2 2 3 3 2 3 1 2 2 3 3 2 2 3 2 3 4 3 1 4 1
2 0 3 3 2 2 1 3 1 3 2 2 4 2 3 3 2 3 1 1 1 3 2 2 2 1 2 3 3 2
2 2 0 1 3 2 2 2 3 3 1 3 1 1 1 2 2 2 3 3 2 2 3 2 2 3 3 1 2 3
1 2 3 0 2 2 2 2 2 3 1 2 2 1 3 2 3 2 2 2 1 1 2 2 2 3 2 2 3 2
2 2 3 1 0 2 2 2 1 3 1 3 3 2 3 3 2 2 2 3 1 2 2 1 2 3 1 3 3 2
2 2 1 2 1 0 2 3 2 2 2 2 2 2 2 2 2 2 3 1 2 2 1 2 3 2 2 2 2 1
1 2 3 2 2 2 0 2 3 3 1 1 3 1 2 2 2 3 3 1 2 2 3 2 2 2 2 2 2 2
2 1 3 3 2 3 1 0 1 1 2 2 3 1 3 2 3 3 2 2 1 2 2 3 2 2 2 3 3 2
1 1 3 2 2 2 2 2 0 2 3 3 4 2 3 2 2 2 2 2 2 3 3 2 1 2 1 2 4 1
2 2 3 3 2 2 1 2 1 0 2 2 2 2 3 3 3 2 2 2 1 1 2 2 1 3 2 3 3 2
2 2 2 2 2 1 2 3 2 3 0 2 2 3 3 3 3 2 2 2 1 1 2 1 3 3 2 3 3 2
1 2 3 1 2 2 2 3 2 2 2 0 3 2 1 2 3 2 2 2 1 2 2 2 3 3 3 2 1 1
2 2 1 1 2 1 3 3 2 3 2 2 0 2 2 2 3 1 3 2 2 2 2 3 3 3 3 2 1 2
1 1 2 2 2 2 2 1 2 2 3 3 3 0 3 1 3 2 2 2 2 2 3 2 1 2 3 2 4 2
2 2 2 1 2 1 1 3 2 3 2 2 3 2 0 3 3 2 2 2 1 2 2 1 3 3 3 3 3 2
2 2 1 2 2 2 2 3 2 4 2 2 2 2 2 0 3 1 2 2 1 1 2 1 3 3 2 2 3 3
2 2 1 2 2 2 2 3 2 4 2 2 2 2 2 0 3 1 2 2 1 1 2 1 3 3 2 2 3 3
2 3 3 2 2 2 2 4 2 2 2 1 3 3 1 3 0 2 2 3 1 2 2 2 3 4 3 1 2 1
2 2 1 2 2 1 3 3 1 3 2 1 2 2 2 1 2 0 3 2 2 2 2 2 2 3 2 1 2 2
2 1 2 2 1 1 2 3 2 2 2 3 3 3 3 3 3 2 0 2 1 3 2 1 3 2 2 3 3 1
2 2 3 2 1 2 2 2 2 2 2 2 4 1 2 2 1 2 2 0 2 3 2 2 2 1 1 2 3 1
1 2 3 2 1 2 1 3 2 3 1 2 3 2 3 2 3 3 1 2 0 2 1 2 3 3 2 2 2 2
2 2 2 2 2 2 3 2 1 3 3 1 1 2 2 2 2 1 2 1 2 0 3 3 2 2 1 2 2 2
1 1 2 2 3 2 2 3 2 3 3 1 2 3 2 1 3 2 2 2 2 1 0 2 3 2 2 2 1 2
2 1 2 1 3 2 2 3 1 3 1 2 3 2 3 2 3 1 2 2 2 2 3 0 2 2 2 2 3 2
2 2 2 2 1 1 2 1 2 2 2 3 3 1 3 2 3 2 3 2 2 3 2 1 0 3 2 3 3 2
2 2 3 2 2 2 3 2 3 3 3 1 3 2 2 2 2 3 2 3 2 2 1 2 1 0 1 3 2 2
2 1 3 3 2 2 2 1 2 2 3 2 4 1 2 2 1 3 1 2 2 3 3 2 1 2 0 2 3 2
2 3 2 1 2 1 3 3 2 3 1 2 2 2 2 3 1 2 3 2 2 1 2 2 3 3 2 0 3 2
1 1 2 2 2 1 2 3 2 3 2 2 2 2 3 2 2 2 2 1 2 1 2 3 3 2 2 1 0 2
```

```
2 2 2 2 1 1 1 3 1 1 2 2 3 2 3 2 3 1 3 2 2 2 2 2 2 3 2 2 3 0

Obtained Capacity Matrix:
0 6 0 1 2 4 8 18 2 6 0 6 0 2 3 2 8 12 3 9 0 4 0 2 3 8 12 6 4 3
4 0 6 3 6 8 6 24 3 15 4 0 8 2 9 12 12 24 3 5 2 0 4 2 6 4 12 24 9 10
0 4 0 1 3 4 8 12 3 9 0 6 0 1 1 4 8 12 3 9 0 4 0 2 2 6 12 6 2 9
1 2 3 0 4 6 10 14 4 12 1 2 2 0 6 6 15 14 4 8 1 1 2 0 4 9 10 14 6 8
2 6 3 2 0 2 6 10 0 6 1 9 3 4 0 3 6 10 0 6 1 6 2 2 0 3 3 15 0 4
4 8 2 6 1 0 4 12 2 2 4 8 4 6 2 0 4 8 3 1 4 8 2 6 3 0 4 8 2 1
4 12 12 10 6 4 0 4 9 3 4 6 12 5 6 4 0 6 9 1 8 12 12 10 6 4 0 4 6 2
12 8 18 21 10 12 2 0 5 3 12 16 18 7 15 8 6 0 10 6 6 16 12 21 10 8 4 0 15 6
1 3 3 4 0 2 6 10 0 4 3 9 4 4 0 2 6 10 0 4 2 9 3 4 0 2 3 10 0 2
6 10 9 12 4 2 1 6 2 0 6 10 6 8 6 3 3 6 4 0 3 5 6 8 2 3 2 9 6 0
0 4 0 2 2 2 8 18 2 9 0 4 0 3 3 6 12 12 2 6 0 2 0 1 3 6 8 18 3 6
2 0 6 1 6 8 12 24 6 10 4 0 6 2 3 8 18 16 6 10 2 0 4 2 9 12 18 16 3 5
0 4 0 1 2 2 12 18 2 9 0 4 0 2 2 4 12 6 3 6 0 4 0 3 3 6 12 12 1 6
1 1 2 0 4 6 10 7 4 8 3 3 3 0 6 3 15 14 4 8 2 2 3 0 2 6 15 14 8 8
2 6 2 2 0 1 3 15 0 6 2 6 3 4 0 3 9 10 0 4 1 6 2 2 0 3 9 15 0 4
4 8 2 6 2 0 4 12 2 4 4 8 4 6 2 0 6 4 2 2 2 4 4 3 3 0 4 8 3 3
8 18 12 10 6 4 0 8 6 2 8 6 12 15 3 6 0 4 6 3 4 12 8 10 9 8 0 2 6 1
12 16 6 14 10 4 6 0 5 9 12 8 12 14 10 4 4 0 15 6 12 16 12 14 10 12 4 0 10
2 3 2 4 0 1 6 15 0 4 2 9 3 6 0 3 9 10 0 4 1 9 2 2 0 2 6 15 0 2
6 10 9 8 2 2 2 6 4 0 6 10 12 4 4 2 1 6 4 0 6 15 6 8 4 1 1 6 6 0
0 4 0 2 1 4 4 18 2 9 0 4 0 2 3 4 12 18 1 6 0 4 0 2 3 6 8 12 2 6
4 0 4 2 6 8 18 16 3 15 6 0 2 2 6 8 12 8 6 5 4 0 6 3 6 8 6 16 6 10
0 2 0 2 3 4 8 18 2 9 0 2 0 3 2 2 12 12 2 6 0 2 0 2 3 4 8 12 1 6
2 1 2 0 6 6 10 21 2 12 1 2 3 0 6 6 15 7 4 8 2 2 3 0 4 6 10 14 6 8
2 6 2 4 0 1 6 5 0 4 2 9 3 2 0 2 9 10 0 4 2 9 2 2 0 3 6 15 0 4
4 8 6 6 2 0 6 8 3 3 6 4 6 6 2 0 4 12 2 3 4 8 2 6 1 0 2 12 2 2
8 6 12 15 6 4 0 2 6 2 12 12 16 5 6 4 0 6 3 2 8 18 12 10 3 4 0 4 9 2
12 24 12 7 10 4 6 0 10 9 6 16 12 14 10 12 2 0 15 6 12 8 12 14 15 12 4 0 15
1 3 2 4 0 1 6 15 0 6 2 6 2 4 0 2 6 10 0 2 2 3 2 6 0 2 6 5 0 4
6 10 6 8 2 1 1 9 2 0 6 10 9 8 6 2 3 3 6 0 6 10 6 8 4 3 2 6 6 0

For k(6), z_opt(4855) density(0.889655)

Shortest Path Graph:
0 2 2 2 1 3 1 2 2 3 2 2 2 2 2 2 1 2 3 2 1 2 2 2 3 2 1 1 3 2
1 0 2 2 2 2 2 2 1 2 2 2 3 3 1 3 2 2 3 2 2 1 3 2 3 2 2 2 1 2
2 2 0 2 1 2 2 2 1 1 2 2 2 2 3 1 2 1 3 2 2 3 2 2 3 1 2 2 2 2
```

```
1 2 2 0 1 2 2 2 2 2 2 2 2 3 2 2 2 1 1 2 2 2 2 2 1 3 2 2 1 1
2 1 2 1 0 2 2 3 1 2 1 2 2 2 2 1 2 2 2 2 2 2 3 2 2 3 1 2 2 2
1 2 2 2 2 0 2 1 3 2 2 2 1 2 3 1 1 2 3 2 2 2 2 3 1 1 2 2 2 2
2 2 1 3 2 3 0 2 2 2 2 3 2 2 3 1 3 1 4 1 1 3 2 3 2 2 2 1 3 2
2 2 2 1 1 3 2 0 2 1 2 3 2 2 2 1 2 2 2 1 2 1 3 2 2 2 2 3 2 2
1 2 2 2 2 1 2 2 0 1 2 2 2 2 2 2 2 2 3 3 1 2 2 1 2 2 2 2 1 1
2 1 3 2 2 2 2 1 1 0 2 2 3 3 2 2 3 3 3 2 2 2 3 1 3 2 1 3 2 2
2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 1 1 1 3 1 2 2 2 3 2 2 2 1 1 1
2 2 1 3 2 3 2 2 2 1 3 0 2 2 1 1 3 2 4 2 2 3 2 2 1 2 2 1 3 2
2 2 2 2 3 1 1 2 3 3 2 2 0 3 2 2 2 2 3 1 2 1 2 2 2 1 2 2 2 1
2 2 2 1 2 2 3 2 3 2 2 1 2 0 1 2 3 2 2 3 3 2 2 3 2 2 3 2 1 1
2 2 2 2 1 2 2 3 2 2 2 1 2 3 0 2 2 1 3 2 3 3 2 3 2 1 2 2 2 1
2 1 1 2 1 2 1 3 2 2 2 2 1 1 2 0 3 2 3 2 1 2 3 3 3 2 2 2 2 2
2 2 2 2 2 2 2 1 2 2 1 2 1 3 2 2 0 1 3 2 3 2 2 3 3 1 3 2 2 1
2 2 2 2 2 2 1 2 1 1 2 2 2 3 2 2 2 0 3 1 2 3 1 2 2 2 2 1 2 1
2 3 1 2 2 2 2 2 1 2 2 2 2 3 2 2 1 1 0 2 2 1 2 1 3 2 3 2 2 2
2 1 3 2 2 3 2 1 2 2 1 2 3 3 2 2 2 2 3 0 2 2 2 2 1 1 1 2 2 2
2 1 1 2 1 2 2 3 1 2 2 2 1 1 2 2 3 1 3 2 0 2 2 2 3 2 2 2 2 2
3 2 1 2 2 2 3 2 2 2 1 2 2 3 1 2 1 2 3 1 2 0 2 1 2 2 2 2 2 1
2 1 3 1 2 1 2 2 2 1 2 3 1 3 2 2 1 2 2 2 3 2 0 2 2 2 2 3 2 2
2 2 2 1 2 2 3 3 2 2 1 1 2 2 2 2 2 2 2 2 1 3 2 0 2 1 3 2 2 1
2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 1 1 2 1 2 0 2 1 3 2 1
1 2 2 1 2 3 2 2 2 2 2 1 2 3 2 2 1 1 2 2 2 2 2 3 2 0 2 1 1 2
3 2 2 2 1 2 1 3 2 2 2 1 2 3 1 2 2 2 3 2 2 1 2 1 2 2 0 2 3 1
1 1 2 2 1 2 1 3 2 2 2 3 2 2 2 2 2 2 3 2 1 2 1 3 3 3 2 0 2 3
2 2 2 1 2 3 2 1 2 1 1 2 3 3 1 2 2 1 2 2 3 1 2 2 2 2 2 2 0 2
2 1 1 2 2 1 2 2 2 2 3 1 1 3 1 2 2 2 3 2 3 2 1 3 2 2 3 2 2 0

Obtained Capacity Matrix:
0 4 0 2 1 6 4 12 2 9 0 4 0 2 2 4 4 12 3 6 0 4 0 2 3 4 4 6 3 6
2 0 4 2 6 8 12 16 3 10 4 0 6 3 3 12 12 16 9 10 4 0 6 2 9 8 12 16 3 10
0 4 0 2 1 4 8 12 1 3 0 4 0 2 3 2 8 6 3 6 0 6 0 2 3 2 8 12 2 6
1 2 2 0 2 6 10 14 4 8 2 2 2 0 4 6 10 7 2 8 2 2 2 0 2 9 10 14 2 4
2 3 2 2 0 2 6 15 0 4 1 6 2 4 0 1 6 10 0 4 2 6 3 4 0 3 3 10 0 4
2 8 4 6 2 0 4 4 3 2 4 8 2 6 3 0 2 8 3 2 4 8 4 9 1 0 4 8 2 2
8 12 4 15 6 6 0 4 6 2 8 18 8 10 9 2 0 2 12 1 4 18 8 15 6 4 0 2 9 2
12 16 12 7 5 12 4 0 10 3 12 24 12 14 10 4 4 0 10 3 12 8 18 14 10 8 4 0 10
1 6 2 4 0 1 6 10 0 2 2 6 2 4 0 2 6 10 0 6 1 6 2 2 0 2 6 10 0 2
6 5 9 8 4 2 2 3 2 0 6 10 9 12 4 2 3 9 6 0 6 10 9 4 6 2 1 9 4 0
0 4 0 2 2 4 8 12 2 6 0 4 0 2 2 4 6 3 3 0 4 0 3 2 4 8 6 1 3
```

```
4 0 2 3 6 12 12 16 6 5 6 0 4 2 3 4 18 16 12 10 4 0 4 2 3 8 12 8 9 10
0 4 0 2 3 2 4 12 3 9 0 4 0 3 2 4 8 12 3 3 0 2 0 2 2 2 8 12 2 3
2 2 2 0 4 6 15 14 6 8 2 1 2 0 2 6 15 14 4 12 3 2 2 0 4 6 15 14 2 4
2 6 2 4 0 2 6 15 0 4 2 3 2 6 0 2 6 5 0 4 3 9 2 6 0 1 6 10 0 2
4 4 2 6 1 0 2 12 2 2 4 8 2 3 2 0 6 8 3 2 2 8 6 9 3 0 4 8 2 2
8 12 8 10 6 4 0 2 6 2 4 12 4 15 6 4 0 2 9 2 12 12 8 15 9 2 0 4 6 1
12 16 12 14 10 8 2 0 5 3 12 16 12 21 10 8 4 0 15 3 12 24 6 14 10 8 4 0 10 3
2 9 1 4 0 2 6 10 0 4 2 6 2 6 0 2 3 5 0 4 2 3 2 2 0 2 9 10 0 4
6 5 9 8 4 3 2 3 4 0 3 10 9 12 4 2 2 6 6 0 6 10 6 8 2 1 1 6 4 0
0 2 0 2 1 4 8 18 1 6 0 4 0 1 2 4 12 6 3 6 0 4 0 2 3 4 8 12 2 6
6 0 2 2 6 8 18 16 6 10 2 0 4 3 3 8 6 16 9 5 4 0 4 1 6 8 12 16 6 5
0 2 0 1 2 2 8 12 2 3 0 6 0 3 2 4 4 12 2 6 0 4 0 2 2 4 8 18 2 6
2 2 2 0 4 6 15 21 4 8 1 1 2 0 4 6 10 14 4 8 1 3 2 0 4 3 15 14 4 4
2 3 2 4 0 2 6 10 0 4 2 6 2 4 0 3 6 10 0 2 1 6 1 4 0 2 3 15 0 2
2 8 4 3 2 0 4 8 2 2 4 4 4 9 2 0 2 4 2 2 4 8 4 9 2 0 4 4 1 2
12 12 8 10 3 4 0 6 6 2 8 6 8 15 3 4 0 4 9 2 8 6 8 5 6 4 0 4 9 1
6 8 12 14 5 8 2 0 10 6 12 24 12 14 10 8 4 0 15 6 6 16 6 21 15 12 4 0 10 9
2 6 2 2 0 3 6 5 0 2 1 6 3 6 0 2 6 5 0 4 3 3 2 4 0 2 6 10 0 4
6 5 3 8 4 1 2 6 4 0 9 5 3 12 2 2 2 6 6 0 9 10 3 12 4 2 3 6 4 0

For k(7), z_opt(4497) density(0.889655)

Shortest Path Graph:
0 2 1 2 2 2 2 2 3 2 3 1 2 2 2 3 3 3 2 1 1 2 2 2 2 3 2 1 2 3
1 0 1 2 2 2 2 3 3 2 1 2 2 2 3 2 2 3 2 1 2 2 1 3 2 3 1 2 2 1
1 3 0 2 1 1 2 2 2 2 2 1 3 2 2 3 3 2 1 2 2 1 2 2 2 3 2 2 2 2
2 2 3 0 2 3 2 2 2 2 2 1 2 2 2 1 3 2 1 2 1 1 2 2 2 3 2 1 2 3
2 2 2 1 0 1 3 1 1 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 1
1 3 2 1 1 0 3 2 2 1 2 2 3 1 3 2 3 3 2 2 2 2 1 3 2 2 2 2 2 2
2 2 1 1 2 1 0 3 2 2 3 2 3 2 1 1 3 3 1 3 2 2 1 3 2 3 2 2 2 2
1 2 2 2 1 2 2 0 2 3 1 2 2 2 2 3 1 2 1 2 1 1 2 2 2 2 2 1 2 2
2 3 2 2 1 1 3 2 0 2 2 2 2 1 2 3 1 1 2 2 2 1 2 2 2 2 2 3 1 2
3 2 3 1 2 3 2 3 3 0 2 2 2 3 2 2 3 2 2 1 1 1 3 2 1 4 2 2 2 2
2 3 2 2 1 1 3 2 2 2 0 2 1 1 3 3 3 3 1 2 2 2 1 3 2 2 2 3 2 2
1 2 2 1 2 2 1 1 3 3 2 0 2 1 2 2 2 3 2 2 1 2 2 2 2 2 3 1 2 2
1 3 2 1 2 2 2 1 2 2 1 1 0 2 3 2 2 3 1 2 2 2 2 3 1 3 2 2 1 2
2 2 1 2 2 2 2 2 3 2 1 2 2 0 2 3 2 2 2 1 1 2 2 3 1 1 2 2 1 1
2 1 2 1 1 1 3 2 2 2 2 2 3 2 0 1 3 3 2 2 2 2 2 3 3 3 2 2 2 2
2 2 2 2 2 2 2 2 3 1 2 1 3 1 1 0 2 3 2 2 2 2 2 2 2 2 1 1 1 2
2 2 3 2 2 2 3 1 1 3 2 2 1 2 1 2 0 2 2 2 1 2 3 3 2 1 3 2 2 2
```

```
1 2 2 2 2 3 3 3 3 2 2 2 2 1 3 2 2 0 2 2 2 2 1 3 2 2 1 2 1 1
1 3 2 1 2 2 3 3 1 3 2 2 2 2 2 2 2 2 0 2 1 1 3 2 1 3 2 2 2 2
2 1 2 2 1 2 1 2 2 1 2 1 3 2 2 2 2 3 2 0 2 2 2 2 2 3 1 1 2 2
2 2 3 2 1 2 2 2 2 2 2 1 1 2 1 2 3 3 2 1 0 2 3 3 1 3 2 2 1 2
2 2 3 2 2 2 2 2 3 2 1 1 2 2 1 2 3 1 2 2 2 0 2 1 1 3 2 2 1 2
2 2 1 1 2 2 2 2 2 1 2 1 2 2 3 2 2 3 1 2 2 2 0 3 2 3 1 2 2 1
1 3 2 2 1 2 1 1 2 2 2 2 2 1 2 2 2 3 2 2 2 2 2 0 2 2 2 2 1 1
3 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 2 1 2 2 0 3 1 1 2 1
2 3 2 2 2 1 2 1 2 2 1 1 2 1 2 3 1 1 2 2 2 2 2 3 2 0 2 2 2 1
2 1 1 2 1 2 3 2 2 2 2 2 2 2 2 1 1 3 2 2 1 2 2 3 2 2 0 2 1 2
2 1 2 2 2 2 1 2 3 2 2 2 2 2 2 2 3 4 2 1 1 3 1 1 2 3 2 0 2 2
1 3 2 1 2 2 3 3 2 1 1 2 2 2 2 2 3 2 1 2 1 1 2 2 2 3 3 2 0 3
2 2 1 2 2 2 2 2 3 2 1 2 1 2 3 3 3 3 2 2 2 2 1 2 1 3 2 1 2 0

Obtained Capacity Matrix:
0 4 0 2 2 4 8 12 3 6 0 2 0 2 2 6 12 18 2 3 0 4 0 2 2 6 8 6 2 9
2 0 2 2 6 8 12 24 9 10 2 0 4 2 9 8 12 24 6 5 4 0 2 3 6 12 6 16 6 5
0 6 0 2 1 2 8 12 2 6 0 2 0 2 2 6 12 12 1 6 0 2 0 2 2 6 8 12 2 6
2 2 3 0 4 9 10 14 4 8 2 1 2 0 4 3 15 14 2 8 1 1 2 0 4 9 10 7 4 12
2 6 2 2 0 1 9 5 0 4 2 6 2 2 0 2 6 10 0 4 2 3 2 4 0 2 3 10 0 2
2 12 4 3 1 0 6 8 2 1 4 8 6 3 3 0 6 12 2 2 4 8 2 9 2 0 4 8 2 2
8 12 4 5 6 2 0 6 6 2 12 12 12 10 3 2 0 6 3 3 8 12 4 15 6 6 0 4 6 2
6 16 12 14 5 8 4 0 10 9 6 16 12 14 10 12 2 0 5 6 6 8 12 14 10 8 4 0 10 6
2 9 2 4 0 1 9 10 0 4 2 6 2 2 0 3 3 5 0 4 2 3 2 4 0 2 6 15 0 4
9 10 9 4 4 3 2 9 6 0 6 10 6 12 4 2 3 6 4 0 3 5 9 8 2 4 2 6 4 0
0 6 0 2 1 2 12 12 2 6 0 4 0 1 3 6 12 18 1 6 0 4 0 3 2 4 8 18 2 6
2 0 4 1 6 8 6 8 9 15 4 0 4 1 6 8 12 24 6 10 2 0 4 2 6 8 18 8 6 10
0 6 0 1 2 4 8 6 2 6 0 2 0 2 3 4 8 18 1 6 0 4 0 3 1 6 8 12 1 6
2 2 1 0 4 6 10 14 6 8 1 2 2 0 4 9 10 14 4 4 1 2 2 0 2 3 10 14 2 4
2 3 2 2 0 1 9 10 0 4 2 6 3 4 0 1 9 15 0 4 2 6 2 6 0 3 6 10 0 4
4 8 4 6 2 0 4 8 3 1 4 4 6 3 1 0 4 12 2 2 4 8 4 6 2 0 2 4 1 2
8 12 12 10 6 4 0 2 3 3 8 12 4 10 3 4 0 4 6 2 4 12 12 15 6 2 0 4 6 2
6 16 12 14 10 12 6 0 15 6 12 16 12 7 15 8 4 0 10 6 12 16 6 21 10 8 2 0 5 3
1 9 2 2 0 2 9 15 0 6 2 6 2 4 0 2 6 10 0 4 1 3 3 4 0 3 6 10 0 4
6 5 6 8 2 2 1 6 4 0 6 5 9 8 4 2 2 9 4 0 6 10 6 8 4 3 1 3 4 0
0 4 0 2 1 4 8 12 2 6 0 2 0 2 1 4 12 18 2 3 0 4 0 3 1 6 8 12 1 6
4 0 6 2 6 8 12 16 9 10 2 0 4 2 3 8 18 8 6 10 4 0 4 1 3 12 12 16 3 10
0 4 0 1 2 4 8 12 2 3 0 2 0 2 3 4 8 18 1 6 0 4 0 3 2 6 4 12 2 3
1 3 2 0 2 6 5 7 4 8 2 2 2 0 4 6 10 21 4 8 2 2 2 0 4 6 10 14 2 4
3 6 2 4 0 2 6 10 0 4 2 6 2 4 0 2 6 10 0 2 2 3 2 4 0 3 3 5 0 2
```

```
4 12 4 6 2 0 4 4 2 2 2 4 4 3 2 0 2 4 2 2 4 8 4 9 2 0 4 8 2 1
8 6 4 10 3 4 0 4 6 2 8 12 8 10 6 2 0 6 6 2 4 12 8 15 6 4 0 4 3 2
12 8 12 14 10 8 2 0 15 6 12 16 12 14 10 8 6 0 10 3 6 24 6 7 10 12 4 0 10 6
1 9 2 2 0 2 9 15 0 2 1 6 2 4 0 2 9 10 0 4 1 3 2 4 0 3 9 10 0 6
6 10 3 8 4 2 2 6 6 0 3 10 3 8 6 3 3 9 4 0 6 10 3 8 2 3 2 3 4 0

For k(8), z_opt(4504) density(0.889655)

Shortest Path Graph:
0 1 1 1 2 2 2 2 2 2 1 1 2 1 2 2 2 2 1 2 2 2 2 2 3 2 1 3 2 2
2 0 2 2 1 1 1 2 2 1 1 3 2 2 2 2 1 3 3 2 1 2 2 2 2 2 2 2 2 1
2 2 0 2 2 2 2 2 1 1 1 3 2 2 2 3 2 2 1 2 1 2 1 2 2 2 2 3 2 2
2 2 2 0 2 2 2 2 2 2 2 1 1 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 1
1 2 2 2 0 1 2 2 2 2 1 2 2 2 2 3 2 3 2 2 2 1 2 2 1 2 2 2 1 1
1 2 2 2 2 0 2 2 2 1 1 2 1 2 2 2 2 3 2 2 2 1 2 1 3 1 2 2 2 2
1 2 1 2 1 2 0 2 2 2 1 2 2 2 1 2 2 3 2 1 2 1 2 2 1 3 1 2 2 2
1 2 1 2 2 1 2 0 2 1 2 1 2 2 2 3 2 3 2 1 2 2 1 2 1 2 2 2 2 2
1 2 1 2 2 3 1 2 0 2 2 2 1 1 2 2 2 3 2 2 2 1 2 1 2 2 2 2 1 1
2 1 2 1 1 2 1 1 1 0 2 2 2 2 2 2 1 3 2 2 2 1 2 2 2 1 2 2 2 2
1 2 2 2 2 2 1 1 1 2 0 2 2 2 2 3 3 3 2 2 2 1 1 2 2 2 2 2 2 1
2 2 1 1 2 1 1 1 2 2 2 0 1 2 2 2 1 3 2 2 2 2 2 1 2 2 2 2 2 2
2 1 3 1 2 2 2 2 2 1 1 2 0 2 3 1 2 2 2 2 2 2 2 3 3 2 2 1 1 2
1 2 2 2 1 2 2 2 1 2 2 2 2 0 2 3 2 2 1 2 2 2 1 1 2 1 2 2 2 1
3 2 3 1 2 2 2 2 2 1 2 2 2 2 0 2 1 3 2 2 2 2 1 2 3 2 1 2 1 1
2 2 2 2 2 2 2 2 1 2 1 3 1 2 2 0 2 2 1 2 1 2 1 2 2 2 1 1 2 2
2 1 3 2 2 2 2 2 2 2 1 2 1 1 3 1 0 3 2 3 2 2 1 1 3 2 2 2 1 2
2 1 3 2 2 2 2 2 2 1 2 3 2 1 2 2 1 0 2 1 2 2 2 2 2 2 1 1 2 1
2 2 3 2 1 2 3 3 2 2 2 3 2 2 1 2 1 1 0 2 3 2 2 2 2 1 2 2 2 2
2 1 2 2 2 2 2 1 3 2 2 2 1 3 1 2 2 3 2 0 1 1 2 1 1 3 2 2 2 2
2 2 2 1 1 2 2 1 2 1 2 2 2 2 1 2 2 3 2 1 0 1 2 2 1 2 2 2 2 1
2 2 1 1 2 3 2 2 2 1 1 2 2 2 1 2 2 2 2 2 1 0 2 1 2 2 2 1 2 1
2 1 3 1 2 1 2 3 2 2 2 2 2 2 3 2 2 3 2 2 2 2 0 2 3 2 1 2 1 1
2 1 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 0 2 2 2 2 2 1
1 2 1 2 2 1 2 3 2 2 2 2 1 2 1 2 1 2 2 3 2 1 2 2 0 2 1 1 2 2
2 1 2 2 1 2 2 2 1 1 2 2 2 1 2 3 2 2 2 1 2 2 2 2 2 0 2 1 1 2
2 2 2 1 2 3 1 2 1 1 2 2 2 2 2 1 2 3 2 2 2 2 2 2 2 2 0 2 1 1
2 2 2 2 1 2 1 2 1 1 2 2 2 2 2 2 2 1 3 2 2 1 3 2 2 2 1 0 1 2
2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 1 2 2 2 1 2 2 2 2 2 2 1 0 2
2 2 2 2 2 3 2 2 1 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1 2 1 1 1 1 0
```

```
Obtained Capacity Matrix:
0 2 0 1 2 4 8 12 2 6 0 2 0 1 2 4 8 12 1 6 0 4 0 2 3 4 4 18 2 6
4 0 4 2 3 4 6 16 6 5 2 0 4 2 6 8 6 24 9 10 2 0 4 2 6 8 12 16 6 5
0 4 0 2 2 4 8 12 1 3 0 6 0 2 2 6 8 12 1 6 0 4 0 2 2 4 8 18 2 6
2 2 2 0 4 6 10 14 4 8 2 1 1 0 4 3 10 14 2 4 2 2 2 0 4 3 10 14 4 4
1 6 2 4 0 1 6 10 0 4 1 6 2 4 0 3 6 15 0 4 2 3 2 4 0 2 6 10 0 2
2 8 4 6 2 0 4 8 2 1 2 8 2 6 2 0 4 12 2 2 4 4 4 3 3 0 4 8 2 2
4 12 4 10 3 4 0 4 6 2 4 12 8 10 3 4 0 6 6 1 8 6 8 10 3 6 0 4 6 2
6 16 6 14 10 4 4 0 10 3 12 8 12 14 10 12 4 0 10 3 12 16 6 14 5 8 4 0 10 6
1 6 1 4 0 3 3 10 0 4 2 6 1 2 0 2 6 15 0 4 2 3 2 2 0 2 6 10 0 2
6 5 6 4 2 2 1 3 2 0 6 10 6 8 4 2 1 9 4 0 6 5 6 8 4 1 2 6 4 0
0 4 0 2 2 4 4 6 1 6 0 4 0 2 2 6 12 18 2 6 0 2 0 2 2 4 8 12 2 3
4 0 2 1 6 4 6 8 6 10 4 0 2 2 6 8 6 24 6 10 4 0 4 1 6 8 12 16 6 10
0 2 0 1 2 4 8 12 2 3 0 4 0 2 3 2 8 12 2 6 0 4 0 3 3 4 8 6 1 6
1 2 2 0 2 6 10 14 2 8 2 2 2 0 4 9 10 14 2 8 2 2 1 0 4 3 10 14 4 4
3 6 3 2 0 2 6 10 0 2 2 6 2 4 0 2 3 15 0 4 2 6 1 4 0 2 3 10 0 2
4 8 4 6 2 0 4 8 1 2 2 12 2 6 2 0 4 8 1 2 2 8 2 6 2 0 2 4 2 2
8 6 12 10 6 4 0 4 6 2 4 12 4 5 9 2 0 6 6 3 8 12 4 5 9 4 0 4 3 2
12 8 18 14 10 8 4 0 10 3 12 24 12 7 10 8 2 0 10 3 12 16 12 14 10 8 2 0 10
2 6 3 4 0 2 9 15 0 4 2 9 2 4 0 2 3 5 0 4 3 6 2 4 0 1 6 10 0 4
6 5 6 8 4 2 2 3 6 0 6 10 3 12 2 2 2 9 4 0 3 5 6 4 2 3 2 6 4 0
0 4 0 1 1 4 8 6 2 3 0 4 0 2 1 4 8 18 2 3 0 2 0 2 1 4 8 12 2 3
4 0 2 1 6 12 12 16 6 5 2 0 4 2 3 8 12 16 6 10 2 0 4 1 6 8 12 8 6 5
0 2 0 1 2 2 8 18 2 6 0 4 0 2 3 4 8 18 2 6 0 4 0 2 3 4 4 12 1 3
2 1 2 0 4 6 10 7 4 8 2 2 2 0 4 6 5 14 2 8 1 2 1 0 4 6 10 14 4 4
1 6 1 4 0 1 6 15 0 4 2 6 1 4 0 2 3 10 0 6 2 3 2 4 0 2 3 5 0 4
4 4 4 6 1 0 4 8 1 1 4 8 4 3 2 0 4 8 2 1 4 8 4 6 2 0 4 4 1 2
8 12 8 5 6 6 0 4 3 1 8 12 8 10 6 2 0 6 6 2 8 12 8 10 6 4 0 4 3 1
12 16 12 14 5 8 2 0 5 3 12 16 12 14 10 8 4 0 15 6 12 8 18 14 10 8 2 0 5 6
2 3 2 2 0 2 6 10 0 4 2 3 2 2 0 2 3 10 0 4 1 6 2 4 0 2 6 5 0 4
6 10 6 8 4 3 2 6 2 0 6 10 3 4 4 2 2 6 4 0 6 10 6 4 4 1 1 3 2 0

For k(9), z_opt(4197) density(0.889655)

Shortest Path Graph:
0 2 1 2 2 2 3 2 2 2 2 1 2 1 1 1 2 2 2 1 2 1 1 2 2 1 2 2 1 2
2 0 3 2 2 1 2 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 2 1 2 2 2 2 1
2 2 0 2 1 2 2 1 2 2 2 2 2 1 2 1 1 3 2 2 2 2 2 1 1 1 2 1 2 2
2 2 2 0 1 2 1 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 2 1 2 1 1 1 2 2
2 1 1 2 0 2 2 1 2 2 1 2 2 2 2 1 2 2 2 2 1 2 1 2 1 1 2 2 2 2
```

```
2 2 2 2 3 0 2 1 2 2 1 2 1 2 2 2 2 2 2 2 1 1 2 1 2 1 1 2 2 2
1 2 2 2 1 2 0 2 1 2 1 2 2 1 2 1 1 2 1 2 2 2 1 2 2 2 2 2 2 1
2 2 1 2 2 2 1 0 1 1 2 2 2 2 2 1 2 3 2 2 2 2 1 2 2 1 2 1 2 1
2 1 2 2 1 2 1 2 0 2 2 2 2 1 1 2 2 2 2 1 2 1 1 2 2 2 1 2 2 2
2 1 1 2 2 2 3 2 2 0 2 2 2 2 2 2 2 3 2 2 1 1 2 2 1 1 1 2 1 2
3 2 2 2 2 2 2 1 2 2 0 2 2 1 1 1 1 1 2 1 2 1 2 2 2 2 2 2 1 2
2 2 2 2 2 2 3 2 2 2 2 0 1 2 1 2 1 2 2 2 1 1 1 1 2 2 1 2 2 1
1 1 2 2 2 2 2 2 1 2 2 1 0 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2
3 1 1 2 1 1 2 1 1 2 2 3 2 0 2 1 2 3 2 2 2 2 2 1 2 2 1 2 2 1
2 2 2 2 1 2 2 1 2 1 2 2 2 1 0 2 1 1 1 2 2 1 2 1 2 2 2 1 2 2
2 2 2 2 2 2 3 2 2 2 2 2 2 1 2 0 2 2 3 1 1 2 2 2 1 2 1 2 1 1
2 1 1 1 2 1 2 2 1 2 1 1 1 2 2 2 0 2 2 2 2 2 1 2 1 2 2 2 2 2
2 1 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 0 2 1 1 2 1 2 1 1 2 2 2 2
1 2 2 2 2 2 1 2 2 1 2 1 2 2 2 2 2 2 0 1 1 2 2 2 2 2 2 1 1 1
2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 1 1 2 0 1 1 1 2 2 1 2 2 2 2
1 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 0 1 1 2 2 1 1 1 2 2
2 1 1 2 2 2 3 2 2 2 2 1 2 2 2 1 2 3 2 2 1 0 1 2 1 1 2 2 1 2
2 2 1 1 2 2 2 2 1 1 3 1 1 1 2 2 2 2 3 2 1 1 0 2 2 2 2 2 2 1
2 1 2 1 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 1 0 1 1 2 2 1 2
1 1 2 2 1 2 3 2 2 2 2 1 2 1 1 2 2 2 2 2 2 1 1 2 0 1 1 2 2 2
2 2 2 1 2 2 2 2 2 1 2 2 1 1 1 2 1 2 1 2 1 2 2 1 2 0 2 2 1 2
2 2 1 3 2 2 2 1 2 1 1 2 1 1 2 2 2 2 3 1 2 2 2 2 2 2 0 1 2 1
1 2 2 2 3 1 3 2 2 2 2 1 1 2 2 1 2 2 3 2 1 2 1 1 2 2 2 0 1 2
2 2 2 2 2 2 3 2 2 1 1 2 2 1 2 1 2 2 3 1 2 1 1 2 2 2 2 1 0 2
2 2 2 2 1 3 2 1 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 1 2 2 0

Obtained Capacity Matrix:
0 4 0 2 2 4 12 12 2 6 0 2 0 1 1 2 8 12 2 3 0 2 0 2 2 2 8 12 1 6
4 0 6 2 6 4 12 16 3 10 2 0 4 2 6 4 12 16 3 10 2 0 4 2 3 8 12 16 6 5
0 4 0 2 1 4 8 6 2 6 0 4 0 1 2 2 4 18 2 6 0 4 0 1 1 2 8 6 2 6
2 2 2 0 2 6 5 14 4 4 2 2 2 0 2 6 10 14 4 8 2 2 2 0 4 3 5 7 4 8
2 3 1 4 0 2 6 5 0 4 1 6 2 4 0 1 6 10 0 4 1 6 1 4 0 1 6 10 0 4
4 8 4 6 3 0 4 4 2 2 2 8 2 6 2 0 4 8 2 2 2 4 4 3 2 0 2 8 2 2
4 12 8 10 3 4 0 4 3 2 4 12 8 5 6 2 0 4 3 2 8 12 4 10 6 4 0 4 6 1
12 16 6 14 10 8 2 0 5 3 12 16 12 14 10 4 4 0 10 6 12 16 6 14 10 4 4 0 10 3
2 3 2 4 0 2 3 10 0 4 2 6 2 2 0 2 6 10 0 2 2 3 1 4 0 2 3 10 0 4
6 5 3 8 4 2 3 6 4 0 6 10 6 8 4 2 2 9 4 0 3 5 6 8 2 1 1 6 2 0
0 4 0 2 2 4 8 6 2 6 0 4 0 1 1 2 4 6 2 3 0 2 0 2 2 4 8 12 1 6
4 0 4 2 6 8 18 16 6 10 4 0 2 2 3 8 6 16 6 10 2 0 2 1 6 8 6 16 6 5
0 2 0 2 2 4 8 12 1 6 0 2 0 1 2 4 8 6 2 6 0 4 0 2 2 2 8 12 1 6
```

```
3 1 1 0 2 3 10 7 2 8 2 3 2 0 4 3 10 21 4 8 2 2 2 0 4 6 5 14 4 4
2 6 2 4 0 2 6 5 0 2 2 6 2 2 0 2 3 5 0 4 2 3 2 2 0 2 6 5 0 4
4 8 4 6 2 0 6 8 2 2 4 8 4 3 2 0 4 8 3 1 2 8 4 6 1 0 2 8 1 1
8 6 4 5 6 2 0 4 3 2 4 6 4 10 6 4 0 4 6 2 8 12 4 10 3 4 0 4 6 2
12 8 12 14 10 8 2 0 10 3 12 16 12 14 10 8 4 0 10 3 6 16 6 14 5 4 4 0 10 6
1 6 2 4 0 2 3 10 0 2 2 3 2 4 0 2 6 10 0 2 1 6 2 4 0 2 6 5 0 2
6 10 3 8 4 1 2 6 4 0 6 10 6 8 4 2 1 3 4 0 3 5 3 8 4 1 2 6 4 0
0 4 0 1 2 4 8 12 2 6 0 2 0 2 1 4 8 12 2 6 0 2 0 2 2 2 4 6 2 6
4 0 2 2 6 8 18 16 6 10 4 0 4 2 6 4 12 24 6 10 2 0 2 2 3 4 12 16 3 10
0 4 0 1 2 4 8 12 1 3 0 2 0 1 2 4 8 12 3 6 0 2 0 2 2 4 8 12 2 3
2 1 2 0 4 6 10 14 4 8 2 2 1 0 2 3 10 14 4 8 2 2 1 0 2 3 10 14 2 8
1 3 2 4 0 2 9 10 0 4 2 3 2 2 0 2 6 10 0 4 2 3 1 4 0 1 3 10 0 4
4 8 4 3 2 0 4 8 2 1 4 8 2 3 1 0 2 8 1 2 2 8 4 3 2 0 4 8 1 2
8 12 4 15 6 4 0 2 6 1 4 12 4 5 6 4 0 4 9 1 8 12 8 10 6 4 0 2 6 1
6 16 12 14 15 4 6 0 10 6 12 8 6 14 10 4 4 0 15 6 6 16 6 7 10 8 4 0 5 6
2 6 2 4 0 2 9 10 0 2 1 6 2 2 0 1 6 10 0 2 2 3 1 4 0 2 6 5 0 4
6 10 6 8 2 3 2 3 4 0 6 10 3 8 2 2 2 6 4 0 6 10 3 4 2 1 1 6 4 0

For k(10), z_opt(3992) density(0.889655)

Shortest Path Graph:
0 2 1 2 2 2 2 3 2 2 2 2 2 2 2 1 1 2 2 1 2 2 1 2 2 2 2 1 2 1
2 0 2 2 2 1 2 1 1 2 1 1 2 1 2 2 2 1 2 2 2 2 2 1 2 2 1 2 1 2
2 1 0 1 1 2 2 2 2 2 2 1 1 1 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2
1 2 2 0 2 2 2 1 2 2 1 1 2 1 1 1 2 2 1 2 1 2 2 1 2 2 1 2 2 2
1 1 1 2 0 2 1 2 2 2 2 2 2 2 2 2 1 2 1 1 2 2 1 2 1 1 2 2 2 2
2 1 1 2 2 0 2 1 2 2 1 2 1 2 2 2 2 1 2 2 2 1 2 2 1 2 2 1 1 2
2 1 2 2 2 2 0 1 1 1 2 2 3 2 1 1 2 1 2 2 2 2 2 1 2 1 2 2 2 2
2 1 2 2 1 1 2 0 2 1 2 2 2 2 1 2 2 1 2 1 1 2 1 2 2 2 1 1 2 2
2 2 1 1 2 2 1 2 0 2 2 1 2 2 2 2 1 2 2 1 2 1 2 1 2 1 1 2 2 1
1 1 2 2 2 2 2 1 2 0 2 2 2 2 1 2 2 2 1 2 2 1 2 2 2 1 2 1 2 1
2 2 1 1 1 1 2 2 2 1 0 2 2 1 2 1 2 1 2 2 1 2 2 2 2 2 1 2 2 2
1 1 2 2 1 2 2 2 2 2 1 0 1 1 1 2 2 1 1 2 2 2 2 2 2 2 1 2 1 2
2 2 2 1 2 2 1 2 2 1 1 2 0 1 1 2 2 2 2 3 2 2 2 2 2 1 2 1 1 1
2 2 2 2 1 1 2 2 2 1 2 2 2 0 2 1 2 1 2 2 2 2 2 2 2 1 2 1 1 1
2 2 1 2 2 2 1 2 1 1 2 2 2 1 0 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1
2 2 2 2 1 1 2 2 2 1 1 2 2 1 2 0 1 2 2 2 2 1 1 1 1 2 2 2 2 1
1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 0 1 2 2 2 2 1 1 2 2 1 1 1 2
2 2 2 2 2 2 1 1 2 2 1 2 2 1 1 2 2 0 1 2 1 2 2 2 2 2 1 1 1 2
2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 2 1 2 0 2 1 2 1 1 2 2 2 2 2 2
```

```
2 2 2 1 2 2 2 2 2 1 1 1 2 2 1 2 2 1 2 0 2 1 2 1 1 2 2 2 2 2
2 1 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 1 1 3 0 2 2 2 1 2 1 2 1 2
2 1 2 2 1 2 2 2 2 2 1 1 2 1 2 2 2 1 2 1 1 0 2 2 2 1 2 2 1 2
2 2 1 2 1 1 1 2 2 2 1 1 2 2 1 2 2 1 2 2 2 1 0 2 2 2 2 2 2 2
2 1 2 2 2 2 2 2 2 1 1 2 3 2 1 1 2 2 2 1 2 2 2 0 1 2 2 2 2 2
2 2 1 2 2 2 2 1 2 1 2 2 2 2 1 2 2 2 2 2 2 2 1 1 0 1 2 1 2 1
2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 1 2 2 1 1 1 1 1 0 1 2 2 2
1 2 2 2 2 1 1 2 1 2 2 2 1 2 2 1 2 1 2 2 2 1 2 2 1 2 0 1 2 2
2 2 1 2 2 2 1 2 1 1 2 2 2 1 2 2 1 2 1 2 2 2 1 1 2 2 1 0 2 2
1 1 1 2 2 2 1 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 1 2 1 2 2 1 0 1
2 1 1 1 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 2 1 1 2 2 2 1 1 2 2 0

Obtained Capacity Matrix:
0 4 0 2 2 4 8 18 2 6 0 4 0 2 2 2 4 12 2 3 0 4 0 2 2 4 8 6 2 3
4 0 4 2 6 4 12 8 3 10 2 0 4 1 6 8 12 8 6 10 4 0 4 1 6 8 6 16 3 10
0 2 0 1 1 4 8 12 2 6 0 2 0 1 2 4 4 12 2 6 0 4 0 1 2 4 8 12 2 6
1 2 2 0 4 6 10 7 4 8 1 1 2 0 2 3 10 14 2 8 1 2 2 0 4 6 5 14 4 8
1 3 1 4 0 2 3 10 0 4 2 6 2 4 0 2 3 10 0 2 2 6 1 4 0 1 6 10 0 4
4 4 2 6 2 0 4 4 2 2 2 8 2 6 2 0 4 4 2 2 4 4 4 6 1 0 4 4 1 2
8 6 8 10 6 4 0 2 3 1 8 12 12 10 3 2 0 2 6 2 8 12 8 5 6 2 0 4 6 2
12 8 12 14 5 4 4 0 10 3 12 16 12 14 5 8 4 0 10 3 6 16 6 14 10 8 2 0 10 6
2 6 1 2 0 2 3 10 0 4 2 3 2 4 0 2 3 10 0 2 2 3 2 2 0 1 3 10 0 2
3 5 6 8 4 2 2 3 4 0 6 10 6 8 2 2 2 6 2 0 6 5 6 8 4 1 2 3 4 0
0 4 0 1 1 2 8 12 2 3 0 4 0 1 2 2 8 6 2 6 0 4 0 2 2 4 4 12 2 6
2 0 4 2 3 8 12 16 6 10 2 0 2 1 3 8 12 8 3 10 4 0 4 2 6 8 6 16 3 10
0 4 0 1 2 4 4 12 2 3 0 4 0 1 1 4 8 12 2 9 0 4 0 2 2 2 8 6 1 3
2 2 2 0 2 3 10 14 4 4 2 2 2 0 4 3 10 7 4 8 2 2 2 0 4 3 10 7 2 4
2 6 1 4 0 2 3 10 0 2 2 6 2 2 0 1 3 10 0 4 2 6 2 4 0 2 6 10 0 2
4 8 4 6 1 0 4 8 2 1 2 8 4 3 2 0 2 8 2 2 4 4 2 3 1 0 4 8 2 1
4 12 8 10 6 4 0 4 3 2 8 6 8 10 6 4 0 2 6 2 8 12 4 5 6 4 0 2 3 2
12 16 12 14 10 8 2 0 10 6 6 16 12 7 5 8 4 0 5 6 6 16 12 14 10 8 2 0 5 6
2 6 2 4 0 2 6 10 0 2 2 6 1 2 0 2 3 10 0 4 1 6 1 2 0 2 6 10 0 4
6 10 6 4 4 2 2 6 4 0 3 5 6 8 2 2 2 3 4 0 6 5 6 4 2 2 2 6 4 0
0 2 0 2 2 4 8 12 2 6 0 4 0 1 2 4 8 6 1 9 0 4 0 2 1 4 4 12 1 6
4 0 4 2 3 8 12 16 6 10 2 0 4 1 6 8 12 8 6 5 2 0 4 2 6 4 12 16 3 10
0 4 0 2 1 2 4 12 2 6 0 2 0 2 1 4 8 6 2 6 0 2 0 2 2 4 8 12 2 6
2 1 2 0 4 6 10 14 4 4 1 2 3 0 2 3 10 14 4 4 2 2 2 0 2 6 10 14 4 8
2 6 1 4 0 2 6 5 0 2 2 6 2 4 0 2 6 10 0 4 2 6 1 2 0 1 6 5 0 2
4 8 4 6 2 0 2 8 1 2 4 8 4 6 2 0 2 4 2 2 2 4 2 3 1 0 2 8 2 2
4 12 8 10 6 2 0 4 3 2 8 12 4 10 6 2 0 2 6 2 8 6 8 10 3 4 0 2 6 2
```

```
12 16 6 14 10 8 2 0 5 3 12 16 12 7 10 8 2 0 5 6 12 16 6 7 10 8 2 0 10 6
1 3 1 4 0 2 3 10 0 4 2 3 2 2 0 2 6 10 0 4 2 6 1 4 0 2 6 5 0 2
6 5 3 4 4 2 2 6 4 0 6 10 3 8 4 1 2 3 4 0 3 5 6 8 4 1 1 6 4 0

For k(11), z_opt(3881) density(0.889655)

Shortest Path Graph:
0 2 2 2 2 2 2 1 2 1 2 2 2 1 2 1 2 2 2 1 2 1 2 2 2 1 1 1 2 1
1 0 1 2 1 2 1 1 1 1 2 2 2 2 2 2 2 2 1 1 2 1 2 1 2 2 2 1 2 2
2 2 0 2 1 2 2 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 2 2 1 2 2
1 1 2 0 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 2 2 2 3 1 2 2 1 2
2 2 2 2 0 2 1 1 2 2 2 2 2 1 2 1 2 1 2 3 2 2 1 1 1 2 2 1 1 2
1 2 2 2 2 0 2 2 3 2 1 2 1 2 2 2 3 2 2 2 2 1 1 2 2 1 1 1 1 1
1 1 1 2 2 2 0 2 2 2 2 1 2 1 2 1 2 2 2 2 1 2 2 2 1 1 2 1 3 1
2 1 1 1 1 2 1 0 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 1 1 1
1 2 2 2 1 2 1 1 0 1 2 2 1 2 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2
2 1 2 2 1 1 1 1 2 0 2 1 2 2 1 1 1 2 2 2 1 2 2 2 2 2 2 2 2 1
2 2 2 1 2 2 1 1 2 2 0 2 1 1 2 2 2 2 2 1 2 2 1 2 2 1 1 2 1 2
1 2 2 1 1 2 2 1 2 2 2 0 2 2 2 2 2 2 1 2 2 2 2 1 2 1 1 2 2 1
1 2 2 1 2 1 1 2 2 2 2 1 0 2 1 2 2 2 1 2 2 1 2 1 2 1 2 2 2 1
1 1 2 2 2 2 2 2 2 1 1 2 1 0 2 1 2 1 1 2 1 2 1 1 2 2 2 2 2 2
1 1 1 2 2 2 1 2 2 2 2 2 1 2 0 2 2 2 2 1 2 1 2 2 2 2 1 2 2 1
2 1 2 2 2 2 2 2 2 2 1 2 2 1 1 0 1 2 1 2 2 2 2 2 2 1 2 1 2 2
1 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 0 2 2 1 2 1 2 2 2 2 1 1 2 1
2 2 1 1 1 2 1 2 1 2 2 2 1 2 2 2 2 0 2 2 2 2 2 1 2 2 1 1 1 1
2 2 1 2 1 1 1 2 2 2 2 1 2 2 2 2 1 1 0 1 2 2 2 2 1 2 2 1 2 1
1 2 2 2 2 2 1 2 2 2 2 2 2 1 1 2 1 1 2 0 1 2 2 1 2 2 1 1 2 2
2 2 1 1 2 1 1 1 2 2 1 2 2 2 1 2 2 2 1 2 0 2 2 2 2 2 1 1 2 2
2 2 2 1 1 2 1 1 2 1 2 1 2 2 2 1 2 2 1 2 2 0 2 2 1 2 2 2 1 1
1 2 1 2 2 1 2 2 2 1 1 1 2 1 1 2 2 2 2 2 2 1 0 2 2 2 2 1 2 2
2 1 2 2 2 1 2 2 1 2 1 2 2 1 1 2 2 2 1 2 1 2 2 0 2 1 2 2 1 2
1 2 1 2 1 2 2 1 2 2 1 2 2 1 2 2 2 2 1 2 1 2 1 2 0 1 1 2 2 2
2 2 1 2 1 2 2 1 2 2 2 2 2 2 1 1 2 2 1 2 1 2 1 2 2 0 2 1 2 2
2 1 2 1 2 1 1 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 2 1 0 2 1 2
2 2 2 1 2 1 2 1 2 1 2 2 2 1 2 2 2 1 2 2 2 1 1 2 2 1 1 0 2 2
1 2 2 2 2 2 1 1 2 2 2 1 2 2 2 1 2 3 2 2 1 1 2 2 2 1 1 2 0 1
2 2 2 1 2 2 2 1 2 2 1 2 1 2 1 3 3 2 2 2 2 2 1 1 1 2 2 1 2 0

Obtained Capacity Matrix:
0 4 0 2 2 4 8 6 2 3 0 4 0 1 2 2 8 12 2 3 0 2 0 2 2 2 4 6 2 3
```

```
2 0 2 2 3 8 6 8 3 5 4 0 4 2 6 8 12 16 3 5 4 0 4 1 6 8 12 8 6 10
0 4 0 2 1 4 8 6 1 3 0 4 0 2 2 4 8 12 2 3 0 2 0 2 2 4 8 6 2 6
1 1 2 0 4 3 10 14 4 8 2 2 2 0 2 6 10 7 4 4 2 2 2 0 6 3 10 14 2 8
2 6 2 4 0 2 3 5 0 4 2 6 2 2 0 1 6 5 0 6 2 6 1 2 0 2 6 5 0 4
2 8 4 6 2 0 4 8 3 2 2 8 2 6 2 0 6 8 2 2 4 4 2 6 2 0 2 4 1 1
4 6 4 10 6 4 0 4 6 2 8 6 8 5 6 2 0 4 6 2 4 12 8 10 3 2 0 2 9 1
12 8 6 7 5 8 2 0 5 6 12 16 12 14 10 8 4 0 10 6 12 16 6 7 10 4 4 0 5 3
1 6 2 4 0 2 3 5 0 2 2 6 1 4 0 1 3 10 0 4 2 3 2 4 0 2 6 10 0 4
6 5 6 8 2 1 1 3 4 0 6 5 6 8 2 1 1 6 4 0 3 10 6 8 4 2 2 6 4 0
0 4 0 1 2 4 4 6 2 6 0 4 0 1 2 4 8 12 2 3 0 4 0 2 2 2 4 12 1 6
2 0 4 1 3 8 12 8 6 10 4 0 4 2 6 8 12 16 3 10 4 0 4 1 6 4 6 16 6 5
0 4 0 1 2 2 4 12 2 6 0 2 0 2 1 4 8 12 1 6 0 2 0 1 2 2 8 12 2 3
1 1 2 0 4 6 10 14 4 4 1 2 1 0 4 3 10 7 2 8 1 2 1 0 4 6 10 14 4 8
1 3 1 4 0 2 3 10 0 4 2 6 1 4 0 2 6 10 0 2 2 3 2 4 0 2 3 10 0 2
4 4 4 6 2 0 4 8 2 2 2 8 4 3 1 0 2 8 1 2 4 8 4 6 2 0 4 4 2 2
4 6 4 10 6 4 0 4 3 2 8 12 8 10 6 4 0 4 6 1 8 6 8 10 6 4 0 2 6 1
12 16 6 7 5 8 2 0 5 6 12 16 6 14 10 8 4 0 10 6 12 16 12 7 10 8 2 0 5 3
2 6 1 4 0 1 3 10 0 4 2 3 2 4 0 2 3 5 0 2 2 6 2 4 0 2 6 5 0 2
3 10 6 8 4 2 1 6 4 0 6 10 6 4 2 2 1 3 4 0 3 10 6 4 4 2 1 3 4 0
0 4 0 1 2 2 4 6 2 6 0 4 0 2 1 4 8 12 1 6 0 4 0 2 2 4 4 6 2 6
4 0 4 1 3 8 6 8 6 5 4 0 4 2 6 4 12 16 3 10 4 0 4 2 3 8 12 16 3 5
0 4 0 2 2 2 8 12 2 3 0 2 0 1 1 4 8 12 2 6 0 2 0 2 2 4 8 6 2 6
2 1 2 0 4 3 10 14 2 8 1 2 2 0 2 6 10 14 2 8 1 2 2 0 4 3 10 14 2 8
1 6 1 4 0 2 6 5 0 4 1 6 2 2 0 2 6 10 0 4 1 6 1 4 0 1 3 10 0 4
4 8 2 6 1 0 4 4 2 2 4 8 4 6 1 0 4 8 1 2 2 8 2 6 2 0 4 4 2 2
8 6 8 5 6 2 0 4 6 2 4 12 8 10 6 2 0 4 6 2 8 12 4 10 6 2 0 4 3 2
12 16 12 7 10 4 4 0 10 3 12 16 12 7 10 8 4 0 10 6 12 8 6 14 10 4 2 0 10 6
1 6 2 4 0 2 3 5 0 4 2 3 2 4 0 1 6 15 0 4 1 3 2 4 0 1 3 10 0 2
6 10 6 4 4 2 2 3 4 0 3 10 3 8 2 3 3 6 4 0 6 10 3 4 2 2 2 3 4 0

For k(12), z_opt(3809) density(0.889655)

Shortest Path Graph:
0 2 1 2 2 2 2 2 2 2 1 1 2 2 1 2 1 1 2 1 1 1 2 2 2 2 1 2 2 1
2 0 2 2 2 1 2 2 2 1 2 2 1 1 2 2 2 1 1 1 2 2 2 1 1 2 2 1 2 1
1 1 0 2 2 1 2 1 1 1 1 2 2 1 2 2 2 1 2 1 1 2 2 2 2 2 2 1 1 2
1 1 1 0 2 2 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 1
2 2 2 1 0 1 2 2 2 1 2 2 2 2 1 1 3 1 1 2 2 2 1 2 1 1 1 2 2 2
1 1 2 1 2 0 2 1 1 1 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 2 1 2 1
1 2 2 2 2 2 0 2 1 2 2 2 1 1 2 1 2 2 2 2 1 2 2 1 1 2 1 1 2 2
```

```
1 1 1 1 2 1 2 0 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 2 2 1 2
2 2 2 2 1 2 2 2 0 1 2 2 2 2 2 1 2 2 1 2 2 1 1 1 1 2 1 2 2 2
2 2 2 1 2 2 1 1 2 0 3 2 1 2 2 2 3 1 2 2 2 2 1 2 2 1 2 1 1 2
2 2 2 1 1 1 1 2 2 2 0 2 2 1 2 1 2 1 1 2 2 1 2 1 1 2 2 2 2 2
1 2 1 2 2 1 2 1 2 2 2 0 2 2 2 2 1 1 1 1 1 1 2 2 1 2 1 2 2 2
2 1 2 2 2 1 1 2 1 1 2 1 0 2 3 1 2 1 2 2 2 1 2 2 2 1 2 2 2 1
2 2 2 1 1 2 2 2 2 1 1 1 1 0 1 2 2 2 2 1 1 2 1 1 2 2 2 2 2 1
1 2 2 2 1 2 2 2 1 2 2 2 2 2 0 2 2 2 1 1 2 1 1 2 1 2 1 2 2 2
2 1 2 2 2 1 2 1 1 1 2 1 1 2 2 0 2 1 2 1 2 1 1 2 2 1 2 2 2 2
2 2 2 2 2 2 2 2 2 1 2 1 2 1 1 2 0 2 1 2 2 1 2 1 1 2 1 2 2 2
2 2 1 2 2 1 2 2 2 2 2 2 1 1 2 2 2 0 2 1 1 2 2 1 1 1 3 2 2 1
2 2 2 2 2 1 2 2 2 1 1 2 2 1 2 1 2 1 0 2 2 1 2 1 2 2 1 2 1 2
1 2 2 1 2 2 2 1 2 2 2 2 1 1 1 1 2 2 1 0 2 1 2 1 2 2 2 2 2 2
2 1 1 1 2 2 1 1 2 1 1 2 2 2 1 1 2 2 2 1 0 1 2 2 2 2 2 2 2 1
2 1 1 2 2 2 1 1 2 2 1 2 2 1 2 2 1 2 2 2 1 0 1 2 2 1 2 2 1 2
2 2 1 1 2 1 2 1 1 1 2 2 1 2 3 2 2 2 1 2 2 1 0 1 2 1 2 2 2 2
2 2 2 2 3 2 1 1 2 2 2 1 2 2 2 2 1 2 2 1 1 1 1 0 1 2 2 1 2 1
1 2 2 1 3 2 2 2 2 1 2 2 2 2 2 1 2 1 1 1 2 2 1 2 0 2 2 1 2 2
2 2 3 2 2 1 3 2 1 2 2 2 2 2 2 1 3 2 1 1 2 2 2 2 1 0 2 1 2 1
1 2 2 2 1 1 1 2 1 1 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 0 1 2 2
1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 1 2 1 1 2 2 2 2 2 0 1 2
1 2 2 2 1 2 1 2 2 2 2 1 2 1 1 2 2 2 2 2 1 2 2 2 2 1 1 1 0 1
2 1 2 1 2 1 2 2 2 2 2 2 2 1 2 2 3 2 1 2 2 2 1 2 1 2 2 1 1 0

Obtained Capacity Matrix:
0 4 0 2 2 4 8 12 2 6 0 2 0 2 1 4 4 6 2 3 0 2 0 2 2 4 4 12 2 3
4 0 4 2 6 4 12 16 6 5 4 0 2 1 6 8 12 8 3 5 4 0 4 1 3 8 12 8 6 5
0 2 0 2 2 2 8 6 1 3 0 4 0 1 2 4 8 6 2 3 0 4 0 2 2 4 8 6 1 6
1 1 1 0 4 6 5 7 2 4 2 2 2 0 4 6 10 14 4 8 2 2 1 0 2 6 10 14 2 4
2 6 2 2 0 1 6 10 0 2 2 6 2 4 0 1 9 5 0 4 2 6 1 4 0 1 3 10 0 4
2 4 4 3 2 0 4 4 1 1 4 8 4 6 2 0 4 8 1 2 4 8 4 3 1 0 4 4 2 1
4 12 8 10 6 4 0 4 3 2 8 12 4 5 6 2 0 4 6 2 4 12 8 5 3 4 0 2 6 2
6 8 6 7 10 4 4 0 10 6 12 16 12 14 10 4 4 0 5 6 12 16 12 7 5 8 4 0 5 6
2 6 2 4 0 2 6 10 0 2 2 6 2 4 0 1 6 10 0 4 2 3 1 2 0 2 3 10 0 4
6 10 6 4 4 2 1 3 4 0 9 10 3 8 4 2 3 3 4 0 6 10 3 8 4 1 2 3 2 0
0 4 0 1 1 2 4 12 2 6 0 4 0 1 2 2 8 6 1 6 0 2 0 1 1 4 8 12 2 6
2 0 2 2 6 4 12 8 6 10 4 0 4 2 6 8 6 8 3 5 2 0 4 2 3 8 6 16 6 10
0 2 0 2 2 2 4 12 1 3 0 2 0 2 3 2 8 6 2 6 0 2 0 2 2 2 8 12 2 3
2 2 2 0 2 6 10 14 4 4 1 1 1 0 2 6 10 14 4 4 1 2 1 0 4 6 10 14 4 4
1 6 2 4 0 2 6 10 0 4 2 6 2 4 0 2 6 10 0 2 2 3 1 4 0 2 3 10 0 4
```

```
4 4 4 6 2 0 4 4 1 1 4 4 2 6 2 0 4 4 2 1 4 4 2 6 2 0 4 8 2 2
8 12 8 10 6 4 0 4 6 1 8 6 8 5 3 4 0 4 3 2 8 6 8 5 3 4 0 4 6 2
12 16 6 14 10 4 4 0 10 6 12 16 6 7 10 8 4 0 10 3 6 16 12 7 5 4 6 0 10 3
2 6 2 4 0 1 6 10 0 2 1 6 2 2 0 1 6 5 0 4 2 3 2 2 0 2 3 10 0 4
3 10 6 4 4 2 2 3 4 0 6 10 3 4 2 1 2 6 2 0 6 5 6 4 4 2 2 6 4 0
0 2 0 1 2 4 4 6 2 3 0 4 0 2 1 2 8 12 2 3 0 2 0 2 2 4 8 12 2 3
4 0 2 2 6 8 6 8 6 10 2 0 4 1 6 8 6 16 6 10 2 0 2 2 6 4 12 16 3 10
0 4 0 1 2 2 8 6 1 3 0 4 0 2 3 4 8 12 1 6 0 2 0 1 2 2 8 12 2 6
2 2 2 0 6 6 5 7 4 8 2 1 2 0 4 6 5 14 4 4 1 1 1 0 2 6 10 7 4 4
1 6 2 2 0 2 6 10 0 2 2 6 2 4 0 1 6 5 0 2 2 6 1 4 0 2 6 5 0 4
4 8 6 6 2 0 6 8 1 2 4 8 4 6 2 0 6 8 1 1 4 8 4 6 1 0 4 4 2 1
4 12 8 10 3 2 0 4 3 1 8 6 8 10 3 4 0 4 6 2 8 12 4 10 6 4 0 2 6 2
6 8 12 14 10 8 4 0 10 6 12 16 6 14 10 4 4 0 5 6 6 8 12 14 10 8 4 0 5 6
1 6 2 4 0 2 3 10 0 4 2 3 2 2 0 2 6 10 0 4 1 6 2 4 0 1 3 5 0 2
6 5 6 4 4 1 2 6 4 0 6 10 6 4 4 2 3 6 2 0 6 10 3 8 2 2 2 3 2 0

For k(13), z_opt(3769) density(0.889655)

Shortest Path Graph:
0 2 2 2 2 2 2 2 2 2 1 1 2 1 1 2 2 2 2 1 2 1 2 2 2 1 1 1 1 2
2 0 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 2 2 2 2 2 2 2 1 1 2 1 2 1
2 1 0 1 1 2 2 2 2 1 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 2
1 2 1 0 2 2 3 1 2 2 2 2 2 2 2 1 1 1 1 1 2 1 2 2 2 2 2 1 1 2
2 2 2 2 0 2 1 2 2 1 1 1 2 2 1 1 2 1 2 1 2 2 2 2 2 1 2 2 2 2
2 1 1 2 1 0 1 2 2 2 1 2 2 2 2 2 2 1 2 2 1 1 1 1 1 2 2 1 2 1
1 2 1 2 2 1 0 1 1 2 2 2 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 2 2 1
1 2 2 1 2 2 3 0 1 2 2 1 2 2 1 2 2 1 2 1 1 2 1 1 2 1 2 1 1 2
2 2 2 1 1 2 2 2 0 1 2 2 1 2 2 2 2 1 2 1 2 2 2 2 1 1 2 1 1 2
1 1 1 2 2 2 2 2 1 0 2 2 2 1 2 2 1 1 2 1 1 2 2 2 2 2 1 1 2 2
2 2 1 1 2 1 2 2 1 2 0 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 1 1 2
2 2 2 1 2 1 2 1 1 2 2 0 2 1 1 2 1 2 1 2 1 1 2 2 2 2 1 2 1 2
2 2 1 2 2 2 1 1 2 2 2 1 0 1 1 2 2 1 2 2 1 1 1 2 2 1 2 1 2 1
1 2 2 2 2 2 1 2 1 2 2 2 1 0 2 2 1 1 1 2 2 2 1 2 2 1 2 1 1 2
1 1 1 2 1 2 2 2 2 1 2 2 1 1 0 2 1 2 1 2 2 1 2 2 2 1 2 1 1 2
2 1 1 2 2 1 2 1 2 2 2 2 2 1 1 0 2 2 1 2 2 1 2 1 2 1 2 1 2 1
2 1 2 1 1 2 2 1 1 2 2 1 2 1 2 2 0 2 1 1 2 1 1 2 2 2 2 2 2 1
2 2 2 1 2 2 2 1 1 2 1 1 2 2 1 1 1 0 1 1 2 2 2 2 2 2 1 2 2 1
2 1 1 2 1 2 2 1 2 2 2 2 2 2 2 2 1 1 0 1 1 2 1 2 2 2 1 1 1 2
2 1 2 2 2 2 2 2 2 2 1 1 2 1 2 1 2 2 1 0 2 2 1 1 1 2 1 2 2 1
1 2 2 1 2 2 3 1 2 2 2 2 2 2 1 2 2 1 2 1 0 1 1 2 2 1 2 1 1 2
```

```
2 1 2 2 1 2 2 2 1 2 2 2 1 1 1 2 1 1 2 2 2 0 2 2 2 2 2 1 2 1
2 2 2 1 1 1 2 1 1 2 1 2 2 1 1 1 2 2 2 2 2 2 0 2 1 2 1 2 1 2
2 2 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 2 2 1 1 1 2 0 1 2 1 2 2 1
2 1 1 2 2 2 2 2 1 1 1 2 1 1 1 2 2 1 2 2 2 2 2 2 0 2 2 2 2 2
2 2 2 2 1 2 2 2 1 2 2 1 1 2 2 1 2 2 2 2 1 2 2 1 2 0 2 1 1 1
2 1 1 1 2 2 2 1 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1 1 2 2 0 1 2 2
1 1 2 2 2 1 2 1 1 1 1 2 2 1 1 2 2 2 2 2 1 2 2 1 2 2 2 0 2 1
2 2 1 2 2 2 2 1 1 1 1 2 1 1 2 1 2 2 1 2 1 2 1 2 2 2 1 2 0 1
2 1 2 2 2 2 1 2 1 2 2 2 2 1 2 2 2 1 1 2 2 1 1 1 2 2 2 1 2 0

Obtained Capacity Matrix:
0 4 0 2 2 4 8 12 2 6 0 2 0 1 1 4 8 12 2 3 0 2 0 2 2 2 4 6 1 6
4 0 4 2 6 8 12 16 3 10 2 0 4 1 3 4 6 16 6 10 4 0 4 2 3 4 12 8 6 5
0 2 0 1 1 4 8 12 2 3 0 4 0 2 1 4 8 6 1 6 0 4 0 2 1 4 8 12 2 6
1 2 1 0 4 6 15 7 4 8 2 2 2 0 4 3 5 7 2 4 2 1 2 0 4 6 10 7 2 8
2 6 2 4 0 2 3 10 0 2 1 3 2 4 0 1 6 5 0 2 2 6 2 4 0 1 6 10 0 4
4 4 2 6 1 0 2 8 2 2 2 8 4 6 2 0 4 4 2 2 2 4 2 3 1 0 4 4 2 1
4 12 4 10 6 2 0 2 3 2 8 12 8 10 3 4 0 4 3 2 4 12 4 10 6 4 0 4 6 1
6 16 12 7 10 8 6 0 5 6 12 8 12 14 5 8 4 0 10 3 6 16 6 7 10 4 4 0 5 6
2 6 2 2 0 2 6 10 0 2 2 6 1 4 0 2 6 5 0 2 2 6 2 4 0 1 6 5 0 4
3 5 3 8 4 2 2 6 2 0 6 10 6 4 4 2 1 3 4 0 3 10 6 8 4 2 1 3 4 0
0 4 0 1 2 2 8 12 1 6 0 4 0 2 2 4 8 12 1 3 0 2 0 1 2 4 8 6 1 6
4 0 4 1 6 4 12 8 3 10 4 0 4 1 3 8 6 16 3 10 2 0 4 2 6 8 6 16 3 10
0 4 0 2 2 4 4 6 2 6 0 2 0 1 1 4 8 6 2 6 0 2 0 2 2 2 8 6 2 3
1 2 2 0 4 6 5 14 2 8 2 2 1 0 4 6 5 7 2 8 2 2 1 0 4 3 10 7 2 8
1 3 1 4 0 2 6 10 0 2 2 6 1 2 0 2 3 10 0 4 2 3 2 4 0 1 6 5 0 4
4 4 2 6 2 0 4 4 2 2 4 8 4 3 1 0 4 8 1 2 4 4 4 3 2 0 4 4 2 1
8 6 8 5 3 4 0 2 3 2 8 6 8 5 6 4 0 4 3 1 8 6 4 10 6 4 0 4 6 1
12 16 12 7 10 8 4 0 5 6 6 8 12 14 5 4 2 0 5 3 12 16 12 14 10 8 2 0 10 3
2 3 1 4 0 2 6 5 0 4 2 6 2 4 0 2 3 5 0 2 1 6 1 4 0 2 3 5 0 4
6 5 6 8 4 2 2 6 4 0 3 5 6 4 4 1 2 6 2 0 6 10 3 4 2 2 1 6 4 0
0 4 0 1 2 4 12 6 2 6 0 4 0 2 1 4 8 6 2 3 0 2 0 2 2 2 8 6 1 6
4 0 4 2 3 8 12 16 3 10 4 0 2 1 3 8 6 8 6 10 4 0 4 2 6 8 12 8 6 5
0 4 0 1 1 2 8 6 1 6 0 4 0 1 1 2 8 12 2 6 0 4 0 2 1 4 4 12 1 6
2 2 2 0 2 3 10 14 4 8 1 2 1 0 2 6 10 14 4 4 1 1 2 0 2 6 5 14 4 4
2 3 1 4 0 2 6 10 0 2 1 6 1 2 0 2 6 5 0 4 2 6 2 4 0 2 6 10 0 4
4 8 4 6 1 0 4 8 1 2 4 4 2 6 2 0 4 8 2 2 2 8 4 3 2 0 4 4 1 1
8 6 4 5 6 4 0 2 6 2 8 6 4 10 6 4 0 4 3 2 8 12 4 5 6 4 0 2 6 2
6 8 12 14 10 4 4 0 5 3 6 16 12 7 5 8 4 0 10 6 6 16 12 7 10 8 4 0 10 3
2 6 1 4 0 2 6 5 0 2 1 6 1 2 0 1 6 10 0 4 1 6 1 4 0 2 3 10 0 2
```

```
6 5 6 8 4 2 1 6 2 0 6 10 6 4 4 2 2 3 2 0 6 5 3 4 4 2 2 3 4 0

For k(14), z_opt(3683) density(0.889655)

Shortest Path Graph:
0 2 2 1 1 2 2 1 1 2 1 2 2 1 1 2 2 2 2 2 1 2 1 1 1 1 2 1 2 1
1 0 2 2 1 2 1 2 2 2 2 1 2 1 2 2 2 2 1 2 1 2 1 2 1 1 2 2 1 1
1 2 0 2 1 2 2 1 1 2 2 1 2 2 1 2 2 2 2 1 2 1 1 2 1 2 2 2 2 1
1 2 1 0 2 1 1 1 2 1 1 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 2 2 1 2
2 1 2 2 0 1 1 2 1 1 2 2 2 1 2 2 1 1 2 2 1 3 2 1 2 1 2 2 1 2
1 1 1 1 1 0 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 1 2 2
1 1 1 2 2 1 0 2 2 1 2 2 1 1 2 2 1 2 1 2 2 2 2 1 2 2 1 1 2 2
2 2 2 1 2 1 2 0 2 2 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 2 1 2 1 1
2 2 1 1 1 2 1 1 0 2 1 1 2 1 2 2 2 2 2 2 1 2 2 2 2 1 2 1 2 1
2 2 2 2 2 2 1 2 2 0 1 1 1 2 2 2 2 1 1 1 1 2 1 2 1 2 1 2 1 1
1 1 2 1 2 2 1 2 2 2 0 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 1 1 2
2 2 2 2 2 2 1 2 2 2 1 0 2 2 2 2 1 1 2 1 1 1 1 2 2 2 1 2 1 1
2 2 2 2 1 2 2 1 2 2 2 1 0 1 1 1 2 2 1 2 1 2 2 2 1 2 1 1 1 1
1 2 2 1 1 1 2 2 2 2 1 1 3 0 2 1 2 2 2 2 2 2 1 2 2 2 1 2 1 1
1 2 2 1 2 2 1 2 1 1 2 1 2 1 0 2 1 2 2 1 2 1 2 1 2 2 1 2 2 2
2 1 1 1 2 1 2 2 2 1 2 1 2 1 2 0 1 2 1 2 1 2 2 1 2 2 2 2 2 2
2 2 1 2 1 1 1 2 2 2 3 2 1 2 1 2 0 2 2 2 1 2 1 2 2 2 1 2 2 1
2 2 2 1 1 1 2 2 2 2 1 1 1 2 1 2 1 0 1 2 2 2 1 2 1 2 2 1 2 2
1 1 2 2 2 1 1 1 2 2 2 2 1 2 1 1 1 2 0 1 1 2 2 1 2 2 2 2 2 2
2 1 1 2 1 2 1 1 1 2 2 2 1 2 2 1 2 1 1 0 1 2 1 2 2 2 2 2 2 2
1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 1 1 2 1 1 0 2 2 2 1 2 2 2 1 2
1 1 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 1 1 1 0 1 2 2 2 2 1 1 1
2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2 1 1 1 1 1 2 0 1 1 2 1 1 2 2
1 1 2 1 2 2 1 1 2 1 2 2 1 2 1 2 2 2 2 2 2 2 2 0 2 1 1 1 2 1
2 1 2 1 1 2 2 2 2 1 2 1 2 2 2 2 1 1 1 2 2 2 2 2 0 2 2 2 1 1
1 2 2 1 2 2 2 1 2 1 1 1 2 1 2 2 2 1 2 2 1 2 2 2 1 0 2 1 1 1
2 1 2 1 1 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 2 1 2 0 1 2 1
1 1 2 2 2 2 2 2 1 1 2 2 2 2 2 2 1 2 2 1 1 3 1 1 1 1 2 0 2 2
2 1 2 2 1 2 2 2 2 1 1 1 2 2 2 1 2 2 1 1 2 2 2 1 2 2 1 1 0 1
2 2 1 2 1 1 2 1 1 1 2 2 2 1 2 1 2 2 2 2 1 2 2 2 2 1 1 2 1 0

Obtained Capacity Matrix:
0 4 0 1 1 4 8 6 1 6 0 4 0 1 1 4 8 12 2 6 0 4 0 1 1 2 8 6 2 3
2 0 4 2 3 8 6 16 6 10 4 0 4 1 6 8 12 16 3 10 2 0 2 2 3 4 12 16 3 5
0 4 0 2 1 4 8 6 1 6 0 2 0 2 1 4 8 12 2 3 0 2 0 2 1 4 8 12 2 3
```

1 2 1 0 4 3 5 7 4 4 1 1 2 0 4 6 10 14 2 8 2 1 2 0 4 6 10 14 2 8
2 3 2 4 0 1 3 10 0 2 2 6 2 2 0 2 3 5 0 4 1 9 2 2 0 1 6 10 0 4
2 4 2 3 1 0 2 4 2 2 4 8 4 6 2 0 2 8 2 2 4 8 2 6 2 0 4 4 2 2
4 6 4 10 6 2 0 4 6 1 8 12 4 5 6 4 0 4 3 2 8 12 8 5 6 4 0 2 6 2
12 16 12 7 10 4 4 0 10 6 12 16 12 14 5 4 2 0 10 6 12 16 6 7 5 8 2 0 5 3
2 6 1 2 0 2 3 5 0 4 1 3 2 2 0 2 6 10 0 4 1 6 2 4 0 1 6 5 0 2
6 10 6 8 4 2 1 6 4 0 3 5 3 8 4 2 2 3 2 0 3 10 3 8 2 2 1 6 2 0
0 2 0 1 2 4 4 12 2 6 0 2 0 2 2 4 4 12 1 3 0 4 0 2 1 4 4 6 1 6
4 0 4 2 6 8 6 16 6 10 2 0 4 2 6 8 6 8 6 5 2 0 2 2 6 8 6 16 3 5
0 4 0 2 1 4 8 6 2 6 0 2 0 1 1 2 8 12 1 6 0 4 0 2 1 4 4 6 1 3
1 2 2 0 2 3 10 14 4 8 1 1 3 0 4 3 10 14 4 8 2 2 1 0 4 6 5 14 2 4
1 6 2 2 0 2 3 10 0 2 2 3 2 2 0 2 3 10 0 2 2 3 2 2 0 2 3 10 0 4
4 4 2 3 2 0 4 8 2 1 4 4 4 3 2 0 2 8 1 2 2 8 4 3 2 0 4 8 2 2
8 12 4 10 3 2 0 4 6 2 12 12 4 10 3 4 0 4 6 2 4 12 4 10 6 4 0 4 6 1
12 16 12 7 5 4 4 0 10 6 6 8 6 14 5 8 2 0 5 6 12 16 6 14 5 8 4 0 10 6
1 3 2 4 0 1 3 5 0 4 2 6 1 4 0 1 3 10 0 2 1 6 2 2 0 2 6 10 0 4
6 5 3 8 2 2 1 3 2 0 6 10 3 8 4 1 2 3 2 0 3 10 3 8 4 2 2 6 4 0
0 2 0 1 2 2 8 12 2 6 0 4 0 2 2 2 4 12 1 3 0 4 0 2 1 4 8 12 1 6
2 0 4 2 6 8 12 16 6 10 4 0 4 1 6 8 12 8 3 5 2 0 2 2 6 8 12 8 3 5
0 4 0 2 2 4 4 12 2 6 0 4 0 2 1 4 4 6 1 3 0 4 0 1 1 4 4 6 2 6
1 1 2 0 4 6 5 7 4 4 2 2 1 0 2 6 10 14 4 8 2 2 2 0 4 3 5 7 4 4
2 3 2 2 0 2 6 10 0 2 2 3 2 4 0 2 3 5 0 4 2 6 2 4 0 2 6 10 0 2
2 8 4 3 2 0 4 4 2 1 2 4 4 3 2 0 4 4 2 2 2 8 4 6 1 0 4 4 1 1
8 6 8 5 3 2 0 4 6 2 8 12 8 10 6 4 0 4 6 1 8 6 8 10 3 4 0 2 6 1
6 8 12 14 10 8 4 0 5 3 12 16 12 14 10 8 2 0 10 3 6 24 6 7 5 4 4 0 10 6
2 3 2 4 0 2 6 10 0 2 1 3 2 4 0 1 6 10 0 2 2 6 2 2 0 2 3 5 0 2
6 10 3 8 2 1 2 3 2 0 6 10 6 4 4 1 2 6 4 0 3 10 6 8 4 1 1 6 2 0

For k(15), z_opt(3704) density(0.889655)