Analysis of Algorithms WS 2022

Prof. Dr. P. Rossmanith
M. Gehnen, H. Lotze, D. Mock

Date: November 14, 2022

Exercise Sheet with solutions 03

Due date: next tutorial session

Tutorial Exercise T3.1

If a flow diagram consists of n nodes and m edges, how many fundamental cycles do we get?

Solution

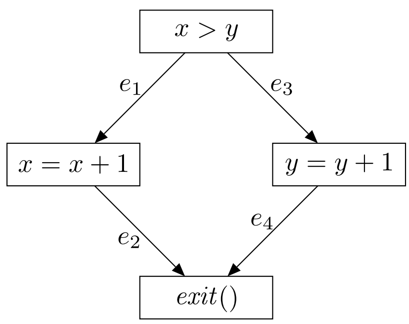
Any spanning tree of a graph on n nodes has to use exactly $n-1$ edges. that means that there are $m-(n-1)$ edges that are not part of the spanning tree. Since every edge not part of the spanning tree is part of exactly one fundamental cycle we get $m-n+1$ many fundamental cycles.

Tutorial Exercise T3.2

Prove or disprove: In every flow diagram you can find a spanning tree such that all fundamental cycles contain only edges that are labeled with plus.

Solution

Consider a part of a program that contains an **if-else**–statement.



A spanning tree of that structure always has one edge not part of the tree, no matter what edge is selected, we always have to use the two edges of the other side in the opposite direction. So without loss of generality let e_1 be the non tree edge, e_2 the edge on the same side (either before or after e_1) and e_3 and e_4 the two edges on the other side. We get

$$C_1 = e_1 + e_2 - e_3 - e_4$$

Which thereby disproves the conjecture.

Tutorial Exercise T3.3

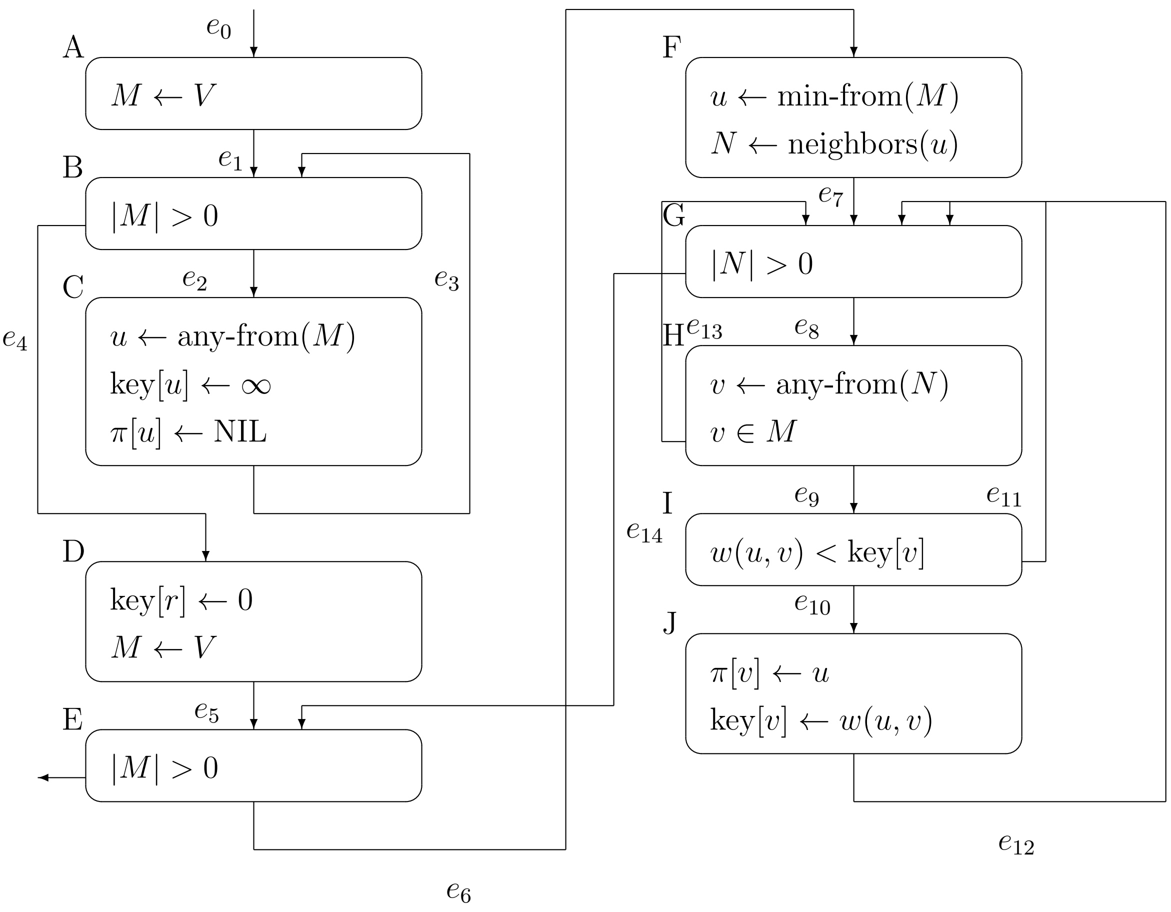
In this exercise, we consider Prim's Algorithm, which computes a minimum spanning tree. The input to this algorithm is a graph $G=(V,E)$, a weight function on the edges $w\colon E \to \mathbf{R}$ and a starting node r.

```
for each u ∈ V do
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
M ← V
while (M ≠ ∅) do
    u ← min-from(M)
    for each v ∈ neighbors(u) do
        if (v ∈ M) ∧ (w(u, v) < key[v]) then
            π[v] ← u
            key[v] ← w(u, v)
```

Construct the control flow graph, a spanning tree in the control flow graph, the fundamental cycles, a corresponding linear system of equations and a solution to this system.

Solution

The flow diagram is depicted below. The **for**-loops were changed, since the initializing and iteration condition must be separated.



We choose the spanning tree $e_1, e_2, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}$. This yields the following fundamental cycles:

$$
\begin{aligned}
C_0 &= e_0 + e_1 + e_4 + e_5 \\
C_3 &= e_3 + e_2 \\
C_{11} &= e_{11} + e_8 + e_9 \\
C_{12} &= e_{12} + e_8 + e_9 + e_{10} \\
C_{13} &= e_{13} + e_8 \\
C_{14} &= e_{14} + e_6 + e_7
\end{aligned}
$$

We now use standard linear algebra to find a good set of blocks whose number of visits we need to compute: By E_i, $0 \leq i \leq 14$, we denote the number of times the program flow visits the edge e_i. With each fundamental cycle above we identify a vector C_i. Then the E_i can be written as a linar combination of the fundamental cycles, i.e.,

$$
(C_0, C_3, C_{11}, C_{12}, C_{13}, C_{14}) \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{pmatrix} = \begin{pmatrix} E_0 \\ E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \\ E_7 \\ E_8 \\ E_9 \\ E_{10} \\ E_{11} \\ E_{12} \\ E_{13} \\ E_{13} \end{pmatrix}
$$

for appropriate values of $\lambda_1, \ldots, \lambda_6$. We select six independent rows and obtain the equation

$$
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{pmatrix} = \begin{pmatrix} E_0 \\ E_2 \\ E_{11} \\ E_{12} \\ E_{13} \\ E_{14} \end{pmatrix} = \begin{pmatrix} 1 \\ C \\ I - J \\ J \\ H - I \\ G - H \end{pmatrix}.
$$

This means, we only need to compute the values of C, G, H, I, J for a complete analysis ($E_0 = 1$ is trivally known), and then all other values can be derived: We see that $E_0 = E_1 = E_4 = E_5$, $E_2 = E_3$, $E_6 = E_7 = E_{14}$, $E_{10} = E_{12}$. This implies $A = 1$, $B = E_1 + E_3 = C + 1$, $D = E_4 = 1$, $E = E_5 + E_{14} = 1 + G - H$, $F = E_6 = G - H$.

Homework Exercise H3.1

Consider the following program:

```
int sel_sort ( int a[], int n ) {
  for ( int i = 0; i < n; ++i ) {
    int min = i;
    for ( int j = i; j < n; ++j ) {
      if ( a[j] < a[min] ) {
```

```
        min = j;
      }
    }
    int temp = a[i];
    a[i] = a[min];
    a[min] = temp;
  }
}
```

The input to this program is an array $a[0, \ldots, n-1]$ that contains n pairwise distinct integer keys in random order.

a) Explain how this program sorts the given array.

b) Analyse how often each instruction of the program is executed on average depending on n.

c) There is only one instruction whose analysis is not trivial. Which one is it?

   Make a table for small values of n by hand that lists the results for this instruction. Compare the table entries with the results from your closed formula that you obtained in b).

Solution

The algorithm is, as indicated by the name, selection sort: it finds the minimal element of the yet-to-be-sorted part and appends it to the sorted part by exchange.

For part b), the outer loop is execute n times, therefore each statement not in the inner loop is executed that often. The if-statement is executed exactly $\frac{n(n+1)}{2}$ times, which leaves the $\min = j$ statement.

The first comparison ($j = i$ to i) always fails, therefore the expected number of executions (cf. Problem T1) then is

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \frac{1}{j-i+1} = \sum_{i=0}^{n-1} \sum_{j=1}^{n-i-1} \frac{1}{j+1} = \sum_{i=0}^{n-1} (H_{n-i} - 1) = (n+1)H_n - 2n.$$

Homework Exercise H3.2

Try to solve the following puzzle: How many subsets of $\{1, \ldots, 2000\}$ have a sum divisible by 5?

Solution

The problem seems to be quite hard to solve with our combinatorial techniques. In a few weeks, you will learn a technique that will make this task pretty simple.