Analysis of Algorithms, WS 2020
Prof. Dr. P. Rossmanith
Dr. E. Burjons, H. Lotze, D. Mock

RWTH AACHEN UNIVERSITY

Date: 4 November 2020

Exercise Sheet with solutions 02

Problem T3

In this warmup exercise we are going to analyse C'—the number of comparisons done in quicksort during partitioning phases *on the left side* of the array. All comparisons are distributed among the following two lines in the program. We split the number of comparisons into C' for the first line and C'' for the second line.

do { $i{+}{+}$; } **while**($a[i] < k$);
do { $j{-}{-}$; } **while**($k < a[j]$);

We would expect C' should be around half the number of comparions, i.e., around $C/2$.

Solution

We get the recurrence

$$C'_N = \frac{1}{N}\sum_{k=0}^{N-1}(C_k + C_{N-1} + k + 1) = \frac{2}{N}\sum_{k=0}^{N-1} C_k + \frac{N+1}{2}$$

because $k+1$ comparisons take place of left side of the pivot element under the conditions that the pivot element will be left at position $k+1$ in the array (when the positions are $1, \ldots, n$). This recurrence is exactly half of the recurrence we had for C_N. Because of linearity the result is then $C'_N = C_N/2$.

Problem T4

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

Solution

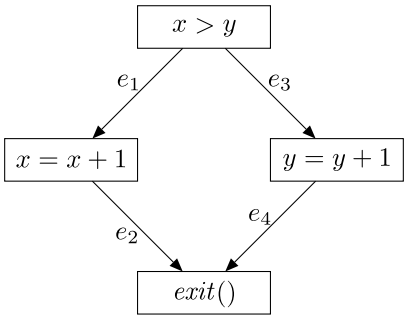
Any spanning tree of a graph on n nodes has to use exactly $n-1$ edges. that means that there are $m-(n-1)$ edges that are not part of the spanning tree. Since every edge not part of the spanning tree is part of exactly one fundamental cycle we get $m-n+1$ many fundamental cycles.

Problem T5

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.

Problem T6

Let $w \in \{a, b\}^n$ a word that has been chosen uniformly at random. How often is the body of the `while`-loop executed on average in the following algorithm? The function `is_palindrome` tests whether a word is a palindrome, i.e., the same when read backwards.

```
i = 2;
while (i <= n) {
  if (is_palindrome(w[1],...,w[i])) return true;
  i++;
}
return false;
```

Solution

Without loss of generality let $w[1]$ be a. If $w[2]$ also is a, which happens with probability $\frac{1}{2}$, we have found a palindrome after one round, and are done, otherwise, we have the prefix ab. In the following round we find a palindrome if an a occurs, otherwise, we have the prefix abb. It follows that we find a palindrome at that moment when a second a occurs. For $1 \leq k \leq n-2$, body is executed k times if and only if $w[2], \ldots, w[k] = b$ and $w[k+1] = a$. The body is executed $n-1$ times if and only if $w[2], \ldots, w[n-1] = b$, as the algorithm terminates after the last round no matter if it found a palindrome or not.

The expected number of iterations therefore is

$$E = \sum_{k=1}^{n-2} \frac{1}{2^k} k + \frac{1}{2^{n-2}}(n-1).$$

We use the fact that

$$\sum_{i=1}^{b} \frac{1}{2^i} = 1 - \frac{1}{2^b}$$

to rewrite the first part of the sum

$$\sum_{k=1}^{n-2} \frac{1}{2^k} k = \sum_{k=1}^{n-2} \sum_{l=1}^{k} \frac{1}{2^k} = \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} [1 \le k \le n-2][1 \le l \le k] \frac{1}{2^k}$$

$$= \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} [1 \le l \le k \le n-2] \frac{1}{2^k} = \sum_{l=1}^{n-2} \sum_{k=l}^{n-2} \frac{1}{2^k} = \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} \sum_{k=1}^{n-l-1} \frac{1}{2^k}$$

$$= \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} (1 - \frac{1}{2^{n-l-1}}) = \frac{n-2}{2^{n-2}} + \sum_{l=1}^{n-2} \frac{1}{2^{l-1}} = \frac{n-2}{2^{n-2}} + 2 - \frac{1}{2^{n-3}}$$

We put both summands toghether to get the final result

$$E = \frac{n-2}{2^{n-2}} + 2 - \frac{1}{2^{n-3}} + \frac{n-1}{2^{n-2}} = 2 - \frac{1}{2^{n-3}} + \frac{1}{2^{n-2}} = 2 - \frac{1}{2^{n-2}}.$$

Problem H4 (15 credits)
We consider the following Algorithm. The array `a` contains a random permutation of the numbers $1, \ldots, N$.

```
void doSomething(int *a, int N)
{
  int i;

  for (i=0; i<N-1; i++)   /*  1  */
    while (a[i] > a[i+1]) /*  2  */
      a[i]--;             /*  3  */
}
```

How often is line 3 executed on average?

Solution
We define the random variable

$$Z_k = \max\{0, a[k] - a[k+1]\},$$

which states how often line 3 is executed when the variable i has the value k. In total, line 3 is executed $Z_1 + Z_2 + \cdots + Z_{N-1}$ times. By linearity of expectation, the expected number of executions is the sum of the expected values of these variables.
For two numbers $x, y \in \{1, \ldots, N\}$ mit $x \neq y$ holds $\Pr[x = a[k], y = a[k+1]] = (N-2)!/N! = 1/N(N-1)$, because there are $N!$ posible permutations and $(N-2)!$ permutations where two values are fixed.
For fixed x and y the third line is executed exactly $\max\{0, x-y\}$ times. This yields the expected value

$$E(Z_k) = \sum_{1 \le y < x \le N} \frac{x-y}{N(N-1)} = \sum_{x=1}^{N} \sum_{y=x+1}^{N} \frac{x-y}{N(N-1)} = \frac{N+1}{6}.$$

By linearity of expectation we get

$$E(Z_1 + \cdots + Z_{N-1}) = \sum_{k=1}^{N-1} E(Z_k) = \frac{(N-1)(N+1)}{6} = \frac{N^2-1}{6}$$

as the expected number of executions.

Problem H5 (15 credits)

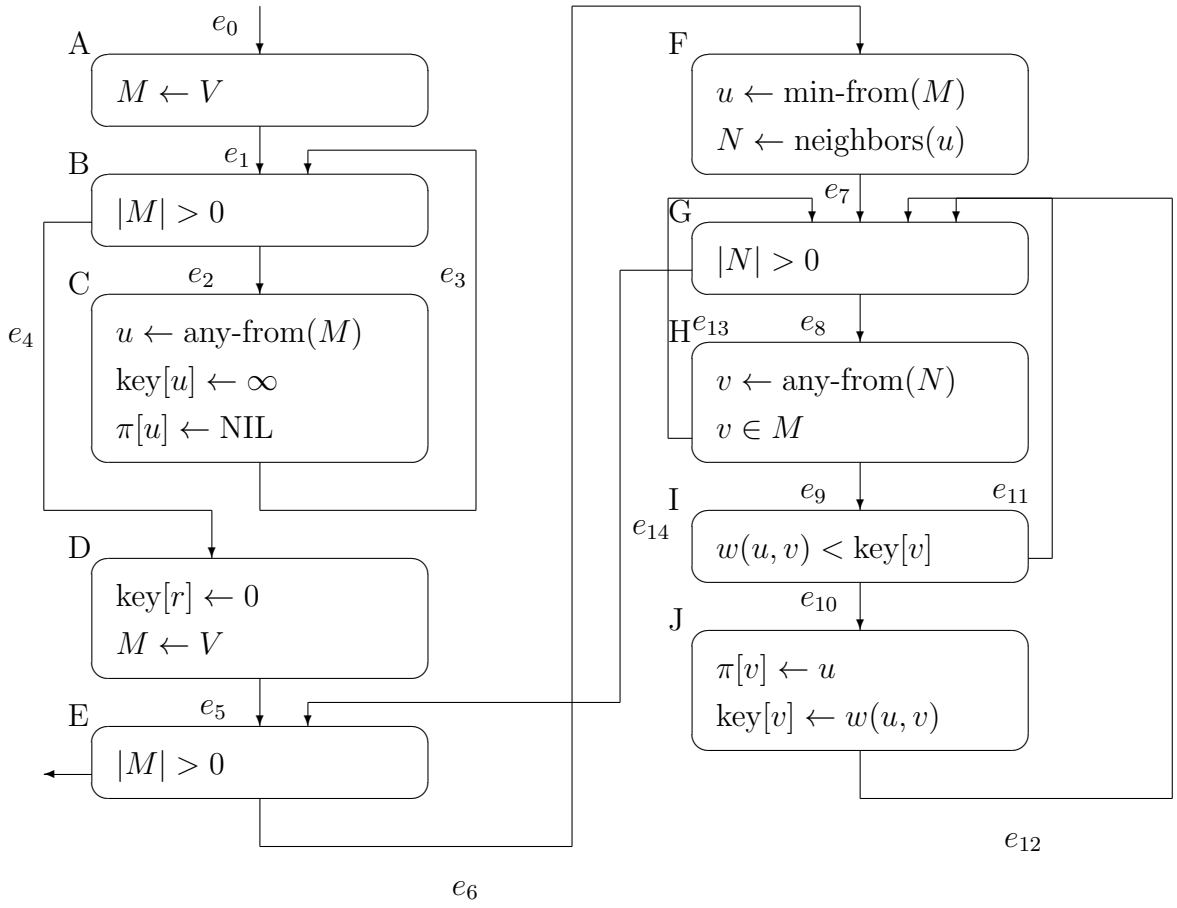
In this exercise, we consider Prim's Algorithm, which computes a minimum spanning tree. The input to this algorithm is a graph $G = (V, E)$, a weight function on the edges $w\colon E \to \mathbf{R}$ and a starting node r.

```
 1  for each u ∈ V do
 2        key[u] ← ∞
 3        π[u] ← NIL
 4  key[r] ← 0
 5  M ← V
 6  while (M ≠ ∅) do
 7        u ← min-from(M)
 8        for each v ∈ neighbors(u) do
 9              if (v ∈ M) ∧ (w(u, v) < key[v]) then
10                    π[v] ← u
11                    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 $A = 1$, $B = E_1 + E_3 = C + 1$, $D = E_4 = 1$, $E = E_5 + E_{14} = 1 + G - H$, $F = E_6 = G - H$.