TÖL101G - Tölvunarfræði 1

Vikublað 4

Lausn

Æfingar

1.4.3

Gerum ráð fyrir að a og b séu fylki af lengd N, þ.e. af taginu `double[]`

```
double sum = 0.0;
// a og b eru af sömu lengd
for (int i = 0; i < a.length; i++) {
  sum += (a[i]-b[i])*(a[i]-b[i]);
}
double distance = Math.sqrt(sum);
```

1.4.15

Gerum ráð fyrir að a sé tvívítt fylki af fylkjum af taginu `int[][]` með N raðir og M dálka og b er fylki með M raðir og K dálka. Þá gefur `a.length` fjölda raða og `a[0].length` fjölda dálka, eins er fjöldi dálka í b gefinn með `b[0].length`. Reyndar gefur `a[i].length` fjölda dálka í röð `i`, en þar sem fylkið er "kassalaga" þá hafa allar raðir jafnmarga dálka.

```
// Fyrir: a[0].length == b.length
int N = a.length;
int M = a[0].length;
int K = b[0].length;

boolean[][] c = new boolean[N][K]; // c is all false initially

for (int i = 0; i < N; i++) {
  for (int j = 0; j < K; j++) {
    // compute c[i][j], initially c[i][j] is false
    for (int k = 0; k < M; k++) {
      c[i][j] = c[i][j] || (a[i][k] && b[k][j]);
    }
  }
}
// Eftir: c = a * b
```

1.4.17

Gerum ráð fyrir að a og b séu fylki af taginu `double[][]`. Við búum til fylkið c til að geyma niðurstöðuna ef fylkjamargföldunin er vel skilgreind.

```
double[][] c; // define the result
```

```
if (a[0].length != b.length) {
  System.out.println("Error, matrix multiplication is not defined");
} else {
  // allocate memory for c
  c = new double[a.length][b[0].length];
  for (int i = 0; i < a.length; i++) {
    for (int j = 0; j < b[0].length; j++) {
      // compute c[i][j], initially c[i][j] = 0.0
      for (int k = 0; k < a[0].length; k++) {
        c[i][j] += a[i][k] * b[k][j];
      }
    }
  }
}
```

Verkefni

Sönnun forrita

Eftirfarandi forritsbútur var gefinn

```
//F : N >=1
int k = 0, n = 1;
while (n <= N/2) {
  //I: n = 2^k og n <= N
  n *= 2;
  k++;
}
// E: 2^k <= N < 2^{k+1}
```

Sanna þarf 3 atriði til að forritið sé rétt.

1. $F \rightarrow I$, þ.e. fyrirskilyrði leiðir til fastayrðingu lykkju
2. $\{I \wedge R\}S\{I\}$, ef fastayrðing lykkju er sönn og við höldum áfram í lykkjunni, þá verður fastayrðing lykkju sönn þegar búið er að keyra lykkjubút
3. $(I \wedge \neg R) \rightarrow E$, ef fastayrðing lykkju er sönn og við hættum í lykkjunni þá er eftirskilyrðið satt

Athugum að ástand forritsins samanstendur af gildunum í breytunum `k,n,N` og því hvar í forriti við erum stödd.

1. Í upphafi er $k = 0, n = 1$ og skv. fyrirskilyrði gildir að $N \geq 1$. Því er $n = 2^k$ og $n = 1 \leq N$ svo að fastayrðing lykkju er uppfyllt.
2. Gerum ráð fyrir að fastayrðing lykkju gildi og R. Þá vitum við að forritið er í ástandi sem uppfyllir $n = 2^k$, $n <= N$ og $n <= N/2$. Þar sem n og k breytast gerum við

greinarmun á fyrir og eftir gildum með n_f, n_e, k_f, k_e. Við sjáum af forritinu að $n_e = 2n_f$ og $k_e = k_f + 1$. Þá höfum við að

$$\begin{aligned} n_e &= 2 \cdot n_f && \text{skv. skilgreiningu á } n_e \\ &= 2 \cdot 2^{k_f} && \text{vegna fastayrðingu lykkju} \\ &= 2^{k_f+1} && \text{algebra} \\ &= 2^{k_e} && \text{skilgreining á } k_e \end{aligned}$$

og $n_e = 2 \cdot n_e \leq 2 \cdot \lfloor \frac{N}{2} \rfloor \leq N$. Svo breyturnar n_e, k_e, N uppfylla fastayrðingu lykkju.

3. Ef fastayrðing lykkju gildir og við hættum í lykkjunni þá gildir. $n = 2^k$, $n \leq N$ og $n > N/2$. Þá gildir líka að $2 \cdot n \geq N$ og því höfum við $n \leq N < 2 \cdot n$. Ef við stingum inn $n = 2^k$ fæst

$$2^k \leq N < 2^{k+1}$$

sem er eftirskilyrðið.

Þar með höfum við sýnt að forritið sé rétt, miðað við fyrir og eftirskilyrði.

Vaxandi stök

```
public class LongestDecreasing {
    public static void main(String[] args) {
        int N = args.length;
        int[] a = new int[N];
        for (int i = 0; i < a.length; i++) {
            a[i] = Integer.parseInt(args[i]);
        }

        // solve problem here
        int j = 0;
        int longest = 1;
        // Pre: j = 0 and a has length >= 1
        for (int i = 1; i < a.length; i++) {
            // Inv: longest is the length of the longest decreasing subsequence
            // of the array a[0],...,a[i-1]
            // and the sequence a[j],...,a[i-1] is decreasing
            if (a[i-1] >= a[i]) {
                if (i-j +1 > longest) {
                    longest = i-j+1;
                }
            } else {
                j = i;
            }
        }
        // Post: longest is the length of the longest decreasing subsequence of a
        System.out.println(longest);
    }
}
```