Correction to

*Producing Good Code for the Case Statement*

Sampath Kannan and Todd A. Proebsting

Department of Computer Science
University of Arizona
Tucson, Arizona 85721 U.S.A.

**Summary**

An $O(n^2)$ algorithm for splitting a case statement's jump table into the minimum number of sub-tables (of a given density) is presented. Previously, the problem was thought to be NP-complete.

KEYWORDS: Case statement, Code generation, NP-complete.

**Algorithm**

In Reference 1, Bernstein asserts that the problem of splitting a case statement's jump table into the minimum number of "clusters" (of a given density) is NP-complete. He states that "this problem is a generalization of the clustering problem given as MS9 in [2]." MS9, however, considers finding the smallest number of clusters with arbitrary distances between pairs of points, whereas this problem is restricted to points that are integers on a line. A simple $O(n^2)$ algorithm that solves this problem follows.

A jump table with a large proportion of *default* entries can be broken into smaller, denser jump tables with a binary or linear search for locating the appropriate smaller table. Bernstein calls these smaller jump tables "clusters."

Let *caseitem* be a sorted array of the $n$ non-default case items. The *density* of a range of case items is the number of non-default entries over the size of the range. The density of the range from $caseitem[i]$ to $caseitem[j]$ (inclusive), $d(i, j)$, is computed as

$$d(i,j) = \frac{j - i + 1}{caseitem[j] - caseitem[i] + 1}$$

For $i = 1$ to $n$, let $minClusters[i]$ be the minimum number of clusters for $caseitem[1]$ through $caseitem[i]$ such that each cluster achieves the desired density, $\theta$. (We define $minClusters[0]$ to be 0.)

$$minClusters[i] = \min_{\substack{0 \le k < i \\ d(k+1,i) \ge \theta}} minClusters[k] + 1$$

Clearly, the above recurrence yields the correct value of $minClusters[1]$. By induction the best clustering of selectors 1 through $i$ is obtained by optimally choosing the cluster $(k+1, \ldots, i)$ — the cluster to contain $i$ — and combining this cluster with the best clustering of 1 through $k$.

The algorithm below computes $minClusters[i]$ according to the above recurrence.

```
minClusters[0] ← 0
forall i ← 1 to n do
    minClusters[i] ← ∞
    forall j ← 0 to i–1 do
        if d(j+1,i) ≥ θ and minClusters[j]+1 < minClusters[i] then
            minClusters[i] ← minClusters[j]+1
        end if
    end forall
end forall
```

The algorithm terminates after computing $minClusters[n]$, the minimum number of legal clusters for all case items. It is trivial to augment the algorithm to indicate the range of each cluster.

Each of the two nested loops iterates at most $n$ times (the number of items). With only a constant amount of additional work per iteration, the algorithm terminates in $O(n^2)$ time.

# References


[1] Robert L. Bernstein. Producing good code for the case statement. *Software—Practice and Experience*, 15(10):1021–1024, October 1985.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.