CSI 5165 Combinatorial Algorithms | Fall 2005
Computer Science | University of Ottawa

Homework Assignment #2 (100 points, weight 15%)
Due: Wednesday, November 9, at 11:30 a.m. (in lecture)

---

1. (25 points) **Backtracking algorithm for all self-avoiding walks**
A *self-avoiding walk* is described by a sequence of edges in the Euclidean plane, beginning at the origin, such that each of the edges is a horizontal or vertical segment of length 1, and such that no point in the plane is visited more than once. There are precisely 4 such walks of length 1, 12 walks of length 2, and 36 walks of length 3.
Define choice sets and describe a backtracking algorithm for the problem of finding **all** self-avoiding walks of length n.

2. (50 points) **Backtracking program for minimum** $2-(v,k,1)$ **covering designs**
A $2-(v,k,1)$ covering design is a collection of k-sets (called blocks) of a v-set such that every pair (2-set) of the v-set occurs in at least one of the k-sets. Our goal is to find coverings with the minimum number of blocks.

   - Describe 2 backtracking algorithms to find a $2-(v,k,1)$ covering design with the **minimum** number of blocks, $C(v,k)$. Algorithm 1 does not use bounding and Algorithm 2 uses bounding.
   - Implement your 2 backtracking algorithms and using each of them, compute $C(v,k)$, for $k = 3,4,5$ and v the largest you can for the given k. For each of your runs, collect statistics on the number of backtracking nodes (BN) and CPU time (T). Summarize your findings on a table with columns:
   $k, v, C(v,k), BN1, BN2, T1, T2$, where the last two pair of values refer to Algorithm 1 and Algorithm 2, respectively.

   Hints:

   (a) Useful global lower bound (Schönheim bound): $C(v,k) \geq \left\lceil \frac{v}{k} \left\lceil \frac{v-1}{k-1} \right\rceil \right\rceil$. Note that you need to develop some "local" lower bound, that is, a lower bound based on a partial solution.

   (b) To know the smallest known covering designs, consult the upper bound table at the "La Jolla Covering repository": http://www.ccrwest.org/cover.html

3. (25 points) **Hill Climbing for embedding of Steiner triple systems**
Develop a hill-clibing algorithm to embed an $STS(w)$ in an $STS(v)$, for $w < v$. In other words, an $STS(w)$ is given, say $(\mathcal{U}, \mathcal{A})$, and we wish to construct an $STS(v)$, say $(\mathcal{V}, \mathcal{B})$ in which $\mathcal{U} \subseteq \mathcal{V}$ and $\mathcal{A} \subseteq \mathcal{B}$.
Hint: Modify the hill climbing algorithm for constructing Steiner Triple Systems given in the textbook. The definition of $STS(v)$ can also be found in the textbook.

Note: For all problems, clarity and efficiency will be taken into account when marking.