SACO 2011 FINALS: DAY 2 SOLUTIONS

Garbage Detection

(problem and solution by Max Rabkin)

This is a complex problem. To get a good score it is likely necessary to use a combination of strategies, and to examine the input files both directly and with analysis programs.

None of our solutions are language-specific; rather, they make use of the redundancy of natural-language text. In any language, some letters are more common than others; only a small set of the infinitely many possible words appear; some phrases and word combinations appear more commonly than others; and so on.

Dictionary techniques. A brief look at the first three files and the last files will reveal that the garbage in them was created letter by letter, and therefore if the garbage fragments contain an English word, it is only by chance. One can therefore use a list of English words to decide which fragments are garbage and which are not.

Note that the nonsense fragments may contain real words (the later test cases are designed to make this more probable by using pairs and triples of letters which commonly appear in English text) and the English fragments may contain words not in the word list (for example, the partial words at the beginning and end of the fragment, names, and spelling errors). Thus the presence or absence of a single word or non-word is not definitive. Some experimentation with thresholds is necessary. Care must also be taken to account for punctuation.

Corpus techniques. After solving the letter-based files, one can create a collection of real English text to work with (such a collection is called a "corpus" in linguistics) by deleting all the garbage fragments from these files. This allows one to compare frequencies of words and phrases in English text with the fragments.

Word frequencies are especially useful in the sixth and ninth files, where the garbage fragments consist of real words, but selected at random with equal probability. Thus words which are very rare in English text appear with equal frequency to very common words like *the* and *and*.

Multigram techniques. In the remaining files, the words are generated with approximately the same frequency as in the source corpus, so frequencies of individual words in the garbage will not be distinguishable from the document fragments.

In the fifth input file, each word is chosen independently. We can detect this form of nonsense by using bigrams—pairs of adjacent words. The frequency of a bigram in a garbage fragment generated this way will be approximately equal to the product of the frequencies of the two words, whereas in natural-language text, some phrases are much more common or less common than this—for example, *the* is a very common word, but *the the* is a very uncommon bigram.

The remaining three files have bigram frequencies which match the source corpus—they were generated by a Markov chain process (see, for example, `http://en.wikipedia.org/wiki/Markov_chain`). For these, we can use a similar algorithm but with trigrams (three adjacent words) instead of bigrams.

To an extent these techniques can be applied without using a corpus, by comparing (say) bigram and trigram frequencies within a single fragment. This approach, particularly, is hampered by the fragments in some of the files being too short.

Compression techniques. We can also take advantage of preexisting algorithms for detecting redundancy. This is essentially what a compression (zipping) algorithm does: it detects redundancy in data and rewrites the data in a less redundant, and therefore smaller, form. So, a file which compresses more is more redundant, and therefore more likely to be natural-language text.

One can use the programs `zip`, `gzip` or `bzip2` for compression; each use different algorithms so may be more or less effective. Alternatively, some of these algorithms may be available in your language's standard libraries.

Auction

(problem and solution by Keegan Carruthers-Smith)

If we treat the toy types as vertices and the bids as edges, we get a graph. In fact the graph we get forms a tree. The problem now can be stated as: given a weighted tree, find the subgraph of the tree which maximises the sum of the edge weights such that the degree of each vertex is $\leq K$. The solution for each of the sub-constraints is:
$K = N$. In this case we can accept every bid, since there are only $N - 1$ bids, but there are N of every toy in stock. So we will never run out of toys.
$N \leq 5$. These constraints are low enough to brute force the problem! You try every combination of bids, if a combination is valid (there is enough stock of toys to satisfy it) we work out the amount of money made. The

answer is then the maximum amount seen. This solution has complexity $O(2^{n-1})$.
$K = 2$. The case of $K = 2$ is an important one, because it leads to the full model solution. Lets look at a single toy, u, and consider all bids involving the toy. Let the bids be $(v_0, c_0), (v_1, c_1), \ldots, (v_m, c_m)$, where (v_i, c_i) represents a bid involving u and v_i for c_i rand.

$$\text{best}_i =$$