Long Runtime and Out-of-Memory Errors

When processing unusually convoluted text, MetaMap may run for a long time and/or, in rare cases, run out of memory. In the latter case, a message such as the following will be displayed to stderr:

```
! Resource error: insufficient memory
```

The most likely explanation for either of these problems is that MetaMap identified so many Candidate Concepts that construction of Final Mappings resulted in a combinatorial explosion, causing physical memory to be exhausted. A typical cause is running MetaMap on poorly screen-scraped bulleted lists.

If you encounter such a problem, we recommend the following:

1. **Disable Composite Phrases**: MetaMap by default creates longer Composite Phrases, which tend to produce better mappings, albeit at the expense of efficiency. Disabling composite phrases using `-Q 0` will result in shorter phrases, and perhaps suboptimal mappings, but often allow MetaMap to complete processing of text that would ordinarily cause an out-of-memory error.

   Example: With composite phrases, the text *pain on the left side of the chest* will be analyzed as a single phrase; without composite phrases, that text will be divided into three smaller phrases, namely [*pain*] [*on the left side*] [*of the chest*].

2. **Prune the Candidate Set**: Each of MetaMap's Final Mappings is a subset of the set of Candidate Concepts; mappings are therefore necessarily exponential in the cardinality of the candidate set. To reduce the number of candidate concepts from which final mappings are constructed, use `--prune N`, where `N` is a positive integer. Typically using 30 or 35 for `N` will allow MetaMap to complete processing without running out of memory. Try progressively smaller values for `N` until MetaMap is able to run to completion.