1 Exercises lectures 1 and 2 (due 29 Jan)

1.1 Hangman: an exercise in entropy and resource scaling

Entropy is a central concept in statistical physics. It is not only useful when dealing with large numbers of particles, but also in other areas dealing with large numbers of something or other. This hangman exercise is an example of that.

Hangman (swedish hänga gubben) is a game where you must guess a word of specific length by guessing letters in it. If you guess too many wrong letters, you are hanged and "die". If you guess a correct letter, the game will reply by telling you where in the word it is. An example transcript of a game:

```
________.
guess e
______e_.
guess n
______e_ (n).
guess s
______e_ (ns).
guess r
______er (ns).
guess i
______er (nsi).
guess o
_o____er (nsi).
guess a
_o____er (nsia).
guess m
_om___er (nsia).
guess computer
The word was: computer.
```

Suppose you are playing a computerised version of hangman. The hangman selects a random word of length w from a dictionary that contains N words in total. You want to create your own hangman player, which has access to the same dictionary, and has to come up with the best letter to guess at every step. This may not be a physics problem exactly, but, because the dictionary is large, it is a computational and statistical problem, and the same things that we consider in this course play a role.

a) Every word (or state) is equally likely to be selected by the hangman computer. Define an entropy S in the same sense as in statistical mechanics, based on the number of possible words n at any particular stage in the game.

b) Entropy is all about information. What does the entropy you have defined in a) tell you about the information that you have about the word?

c) Start simple with $w = 3$ where you have not made any guesses yet. Write down an expression for the expectation value of the change in the entropy $\langle \Delta S \rangle$ when your first guess is "a". You can use the notation of $W(s)$ to denote the number of words that fit a particular value s of the observable. In this case s can take the string values a__,_a_,...aaa,___(a). The latter denotes the state of the observable when there has turned out to be no "a" in the

word. Denote the set of all states of the observable for the next step after ___ that contain an "a" by A(___,a).

d) At every step, you obviously want to guess a letter that will give you a lot of information, thus minimising the entropy, while making as few mistakes as possible. Write down an expression for the average entropy change per mistake, $\langle \Delta S \rangle / \langle m \rangle$. Use $A(s$,b) to describe the set of all possible replies starting from the state of the observable s with, for example, a "b" guessed correctly.

e) Before you could implement this idea, you would also need to know if this is computationally doable. What order of magnitude are the parameters that will play a role in the cpu and memory resource scaling: word length w, dictionary size N, and alphabet length a?

f) How would you implement this? Describe an algorithm explicitely. Try to minimise the number of times you have to search through the dictionary. Make a few simple assumptions about languages that seem reasonable and estimate roughly how the evaluation of the parameter that is to be optimised scales with N, w, and a. As this depends on how you choose to implement the expression found in e), it may not be the same for everyone.

g) Do you think this computation will be doable?

For your information, the algorithm I have in mind was able to correctly guess 46% of words from a dutch dictionary without any mistakes, and 97% with 5 or fewer mistakes. See the plot below.