CS 231
Naomi Nishimura

Calculating worst-case costs

1 Introduction

In order to emphasize expertise in the course material over expertise in Python, assignments and exams may restrict you to using only a subset of the functions built into Python. The information listed here is intended to include all of the allowed operations and functions. Please consult with course personnel if you find yourself wanting to use an operation or function not appearing in this document and not explicitly forbidden.

In the spirit of focusing on algorithm design techniques, the lists of costs given in this document are not intended to reflect the actual costs of Python functions and operations, but rather to allow the use of Python as a type of pseudocode representing a variety of programming languages. Although in many cases the costs given in the document will be accurate estimates of costs of actual Python operations, there will be notable exceptions, especially when using constructs such as Python lists. As lists are not built into most languages, keeping the costs general (that is, reflecting costs typical for most languages) means that the costs may differ substantially from how Python is typically implemented.

2 Mathematical and Boolean operations and functions

Caution: Some of the operations listed in this section are also used on other types of data. The costs given here only apply to numerical and Boolean data. Please see the appropriate sections for other types of data.

In general, the cost of a function or operation will be proportional to the number of inputs. Operations that have two inputs can be executed in constant time. Examples include the following:

- $+$, $-$, $\backslash$, $*$,
- $<$, $>$, $<=$, $>=$, $==$, $!=$
- `and`, `or`, `not`

When a function has multiple inputs, the worst-case cost will be $\Theta(n)$, where n is the number of inputs. Examples of such functions include the following:

- `max`
- `min`

3 Python lists

Although Python provides many built-in list functions, because we are interested in learning how to create and analyze our own solutions, we will tend to use only a limited set of simple list functions. Doing so has the added benefit of our generating algorithms that can be easily translated into other programming languages.

Please use the following worst-case costs for the list operations given below. The value of k is the length of the list on which the operation is executed.

Unless specified otherwise on an assignment or an exam, you should not use any list methods other than those presented here.

- Access or update a particular item in a list using `[]` in $\Theta(1)$ time.
- Create a new empty list in $\Theta(1)$ time
- Create a new list of length k in $\Theta(k)$ time
- Create a list of k items in $\Theta(k)$ time (using `[]` or `list`)
- Copy a list in $\Theta(k)$ time
- `in` in $\Theta(k)$ time (Note: The use of `in` as part of `for item in a_list` has a different cost; in that case, the cost of iteration management is in $\Theta(1)$.)
- Use the slice operation in $\Theta(k)$ time
- Concatenate lists of lengths k and ℓ in $\Theta(k + \ell)$ time (using `+`)
- `len` in $\Theta(1)$ time
- `append` in $\Theta(1)$ time
- Insert an element in an arbitrary location in $\Theta(k)$ time
- `pop` the first or last item in the list in $\Theta(1)$ time
- Delete an item using `del` or `remove` in $\Theta(k)$ time
- Use `map`, `filter`, or `reduce` in time $\Theta(kf(k))$ were $f(k)$ is the cost of a single execution of the function provided
- `sorted` in $\Theta(k \log k)$ time

Remember that you will be computing running time as a function of the size of the input. When k is a constant, then you can assume that an operation that can be executed in $\Theta(k)$ time is a constant-time operation.

4 String, tuples, and dictionaries

How we calculate the costs of string operations and functions will depend on whether or not the length of the string is related to the size of the input to the function. If it is not related to the size of the input, then we can consider all operations and functions to take time in $\Theta(1)$. If instead the length of the string is related to the size of the input, please use the worst-case cost of the analogous list operation.

You are not likely to be analyzing code that uses tuples or dictionaries. If you do use tuple operations, please use the worst-case costs of analogous list operations.

5 User-defined functions, classes, and objects

Although there is no cost incurred by defining a function, there is a cost associated with each application of a function. Similarly, there are costs associated with using class definitions and methods, but not with defining them.

To determine the costs of using a user-defined function or method, use the guidelines given for code as described above. The cost of accessing an attribute can be calculated as $\Theta(1)$ time.

For modules supplied in class, such as `grid.py` and `graph.py`, please see the accompanying documents to find out worst-case running times to use for analysis of various methods and functions.