Analysis of the existing packages on Amazon Glacier

1. Amazon does not actually provide users with their own command line interface (which could have been super nice), but they do support standard API called boto for python. (They also support various languages)

2. People tend to develop their own packages with boto and this includes glacier-cli (https://github.com/basak/glacier-cli.git) and commercial programs like fast-glacier.

3. Fast-glacier was easy to use as it was a commercial one and was in GUI. However there were a few problems with it in that it was impossible to track the amount of requests that the program was making on my account and knowing that the retrieval requests takes up to five hours or so, GUI didn't seem the best solution. After all, the major problem was that I couldn't keep track of retrieval requests and amount of data getting retrieved, which is crucial for billing

4. Glacier-cli had nice ideas floating around as it used git annex as archiving system or keeping track system, while using glacier as storage. However, a major set down would be its annoying setup. Even though I got it working after few tries, it seemed as if the programmer wrote it under the assumption that users are aware of how boto works (credentials and etc). Also integrating git-annex with glacier-cli wasn't easy and I wasn't able to get it working.

   After installing just the cli part, the good feature of glacier-cli was that it had –wait option. It allowed the users to halt its operation until the command is carried out. This was extremely helpful in that many operations in Glacier is delayed up to 4 hours and the only way to know whether the job has finished or not is by subscribing to the job status through E-mail or Amazon SNS. Receiving notification can be more annoying and hard to understand as they are in non-human friendly format. This –wait option has simplified it much nicely. The user can also not use –wait option, so the user doesn't necessarily have to wait for it.

   Also this program does keep records of what has been stored locally, since archive list retrieval is also a delayed operation. These two things are the best functions of this command line interface. Other things have to be tested more implicitly, but every function seems to function seamlessly.

5. One thing that these packages are missing is the billing aspect of the Amazon Glacier. Most of the instructions of these packages emphasized on the importance of understanding billing aspect of it, but failed to incorporate it within the program. This is probably due to the fact that Amazon SNS (Simple Notification Service) has to be incorporated in it. Amazon SNS lets the users to get alerts for various things including job done or billing limit exceeded. Since I'm not acquaint with Amazon SNS, I do not know the limitations and boundaries of this, but it is certain that

users are allowed to set billing alerts in which when estimated charge goes over certain limit that the user sets, the users will be receiving emails. However, this is only an estimated charge, not real-time as delayed requests are present in glacier.

6. Also I've made a mistake last time by telling you, that users are free to retrieve any amount of data as long as it is under free monthly allowance, but it happens to be that it has to be under free daily allowance, which monthly allowance is divided up by the number of days in a month. So basically, about 0.17 percent of the data stored in Glacier every day.

7. In conclusion, there seems not be a way to actually tell the user command by command, how much charge is estimated or what not. The best way to minimize the charges would be to tell the program to chop up the data and retrieve it in a span of a time. Also the user can set up billing alert on their email account, so that they acknowledge the estimated charge and stop the program from doing anymore interaction with Glacier. Basically, billing notification would have to be separated from the uploading or retrieving program. Retrieving program would have to calculate which method would minimize the retrieval fee and carry out such method, but without knowing the actual charge itself.