**蛋白质互相作用最短路径项目报告**

**Motivation**

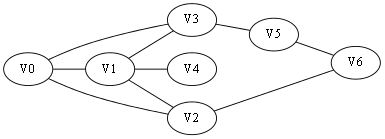
在细胞内，许多蛋白质存在直接互作，如一系列蛋白激酶、蛋白磷酸酯酶及其底物，以及一些蛋白类信号分子及其受体。但是更多的蛋白质并不会在代谢通路内进行直接相互作用，而是通过分别与其它蛋白质互作产生联系。在分子生物学等领域的研究中，我们常常需要确定不直接互作但存在关联的蛋白质之间的关系，以此来研究一些通路的具体作用过程和机理。蛋白质之间最短作用路径的长短可以看作它们生物关联度的衡量标准。因此，我们尝试设计了可以快速给出两个给定蛋白质之间最短作用路径的脚本，用于确定蛋白质的互作关系，从而帮助我们更好地了解不同通路之间的关联，更加精准、高效、严谨地开展相关生物学研究。

**Algorithm**

1. 算法使用：

由于蛋白质之间互作的强度数据难以获取和准确分配蛋白质互作强度的路径权重；并且在运算过程中可以牺牲一部分互作强度误差带来的准确度下降来提高运算速度，我们选择将两个蛋白质直接互作的强度即权重设为1. 因此，我们选择BFS算法进行最短路径的计算。BFS算法在时间复杂度（O( V+E)）上也显著优于其他常用于计算最短路径的算法，如Floyd-Warshall Algorithm, Dijkstra algorithm, Bellman-Ford algorithm等。

BFS即广度优先搜索，是连通图的一种遍历策略。在搜索过程中，所有初始（未搜索）节点是白色，搜索过的节点标记为黑色，从目前所在的点逐一搜索它的子节点，按照标记顺序继续逐一搜索直到所有节点都被标记。



如上图所示，我们从V0开始，V0的子节点包括V1, V2, V3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V0 |  |  |  |  |  |  |  |  |  |
| V0 | V1 | V2 | V3 |  |  |  |  |  |  |

V0访问过后，访问V1，并将它的子节点V4加入下一步搜索的目标点中（V2, V3已加入队列中）。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V0 | V1 | V2 | V3 | V4 |  |  |  |  |  |

接下来继续按照顺序访问节点，直到遍历所有节点。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V0 | V1 | V2 | V3 | V4 | V6 |  |  |  |  |
| V0 | V1 | V2 | V3 | V4 | V6 | V5 |  |  |  |

这样我们就得到了通过BFS遍历连通图的结果。

在具体的算法实现中，我们将已访问过的节点加入visited列表中，搜索完毕后进行回溯，返回路径中的蛋白。

2. 整体思路

（1）在比较了BioGRID和HPRD两个数据库后我们发现，BioGRID数据库的数据使用的是动态加载包，所需的蛋白质信息并不在网页源码中，使用selenium抓取数据十分困难。而HPRD的蛋白质信息全部包含在网页源码中，只要使用selenium的相关方法就可以轻松抓取，因此综合数据获取难度，我们选择了HPRD作为蛋白质互作数据来源。

（2）在获取蛋白质数据的过程中，我们使用了正则表达式来定位相关的蛋白质信息并进行抓取。抓取后我们将蛋白质和其相对应的网址最后几位数字编号进行映射并存储成字典的形式，便于进行调用。

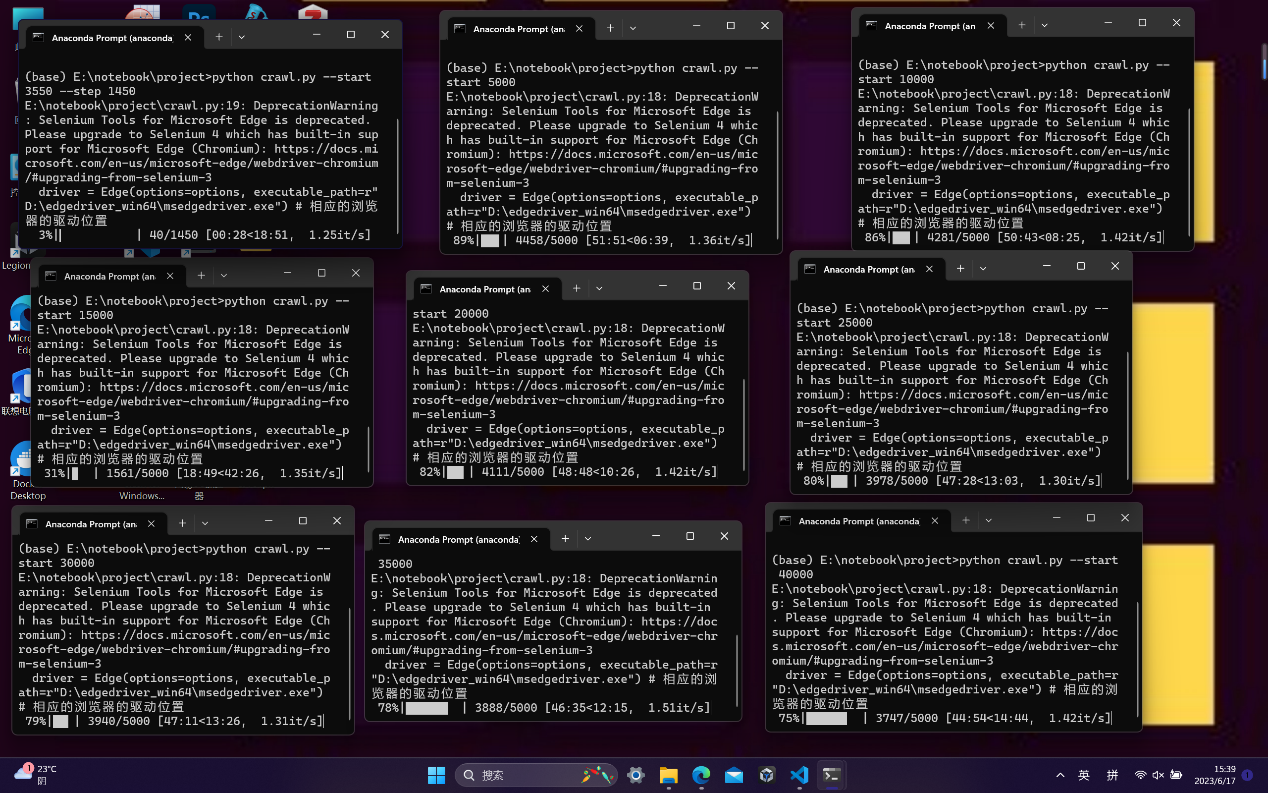
（3）爬取数据时我们发现有的蛋白对应多个名称和编号，而有的蛋白同时与两个蛋白进行互作，我们对于这些情况设置了专门的分类讨论并加以解决。但是，HPRD数据库的蛋白质编号并不连续，也就是说经常出现编号相应的网址为空的情况。我们在尽量保证爬取数据完整性的情况下考量了编号大小与数据密集程度的关系，发现100001以上的编号几乎没有新的蛋白质与之对应，因此我们的编号只爬取到18000。在九个进程同时工作的情况下，爬取数据约耗时5h.

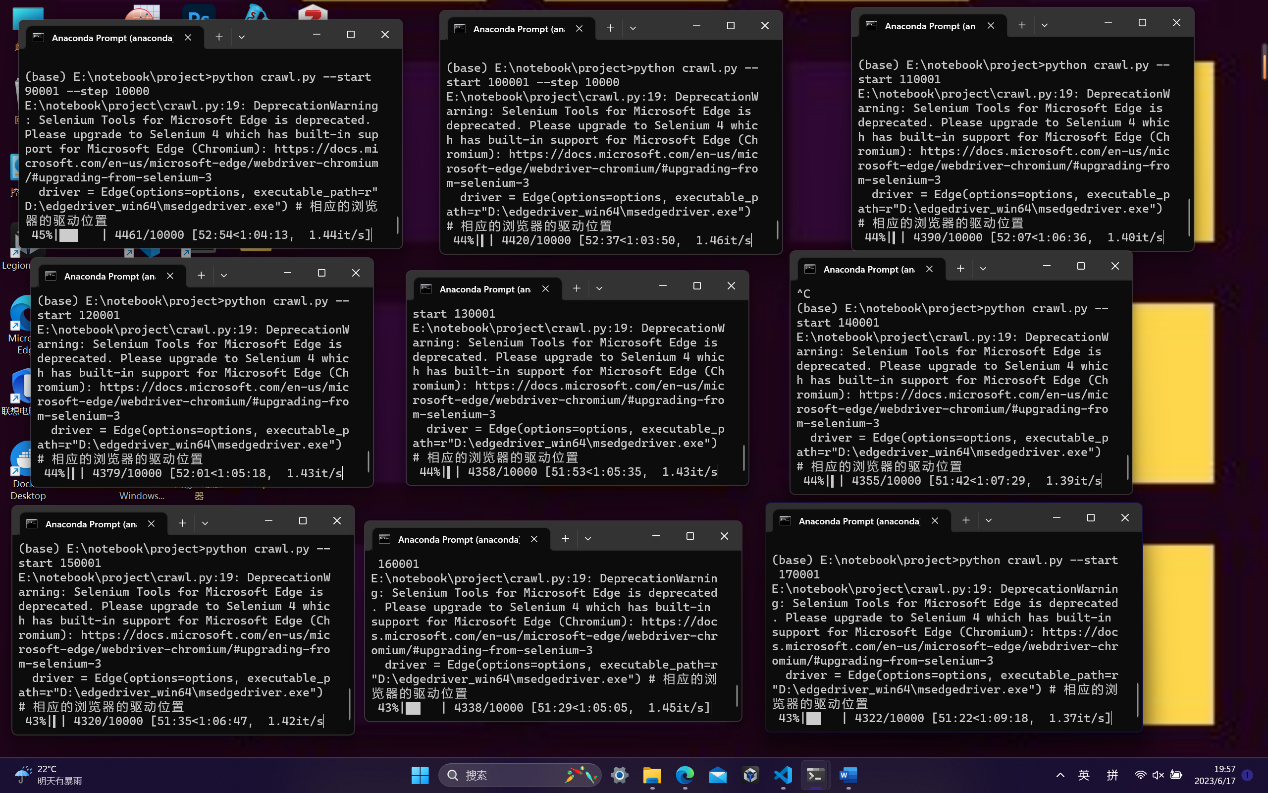
（4）在完成了数据爬取和存储后，我们使用了BFS广度优先搜索算法进行最短路径的查找，在上文已经介绍，这里不加赘述。

**Experiment**

1. 数据爬取：

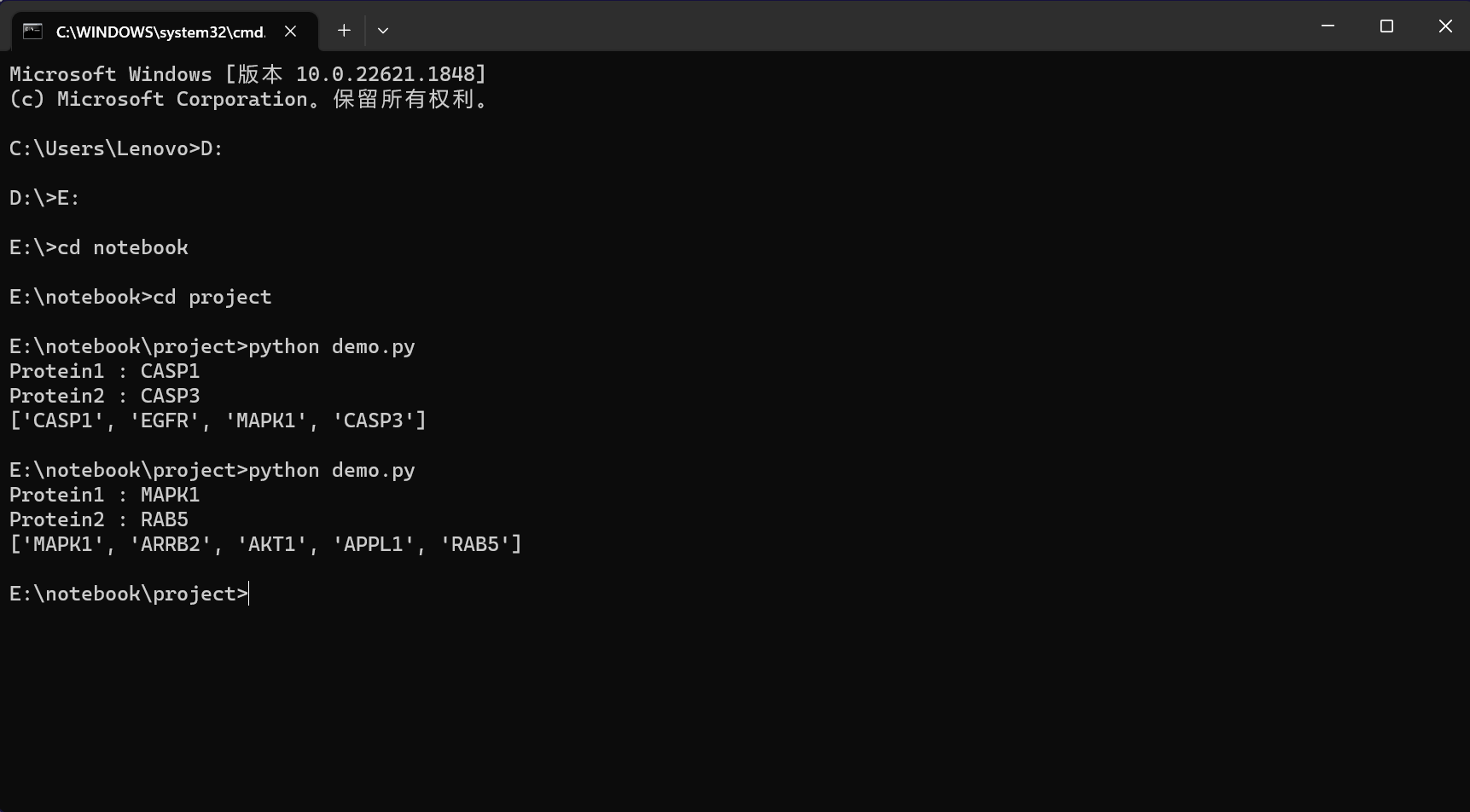
我们使用selenium对HPRD数据库进行数据爬取，找到其中代表蛋白质的信息。由于在该数据库中蛋白质和它们的相应编号存在对应关系，因此可以将蛋白质及其编号做映射并存储。在爬虫过程中，我们选择同时开启多个进程以加快爬取速度。如图为两部分蛋白质数据爬取过程的进度图。





2. 数据测试：

在爬取了一定量的数据之后，我们使用了两组已知蛋白来测试该脚本的运行效果。

可以看到CASP1/CASP3和MAPK1/RAB5给出的最短作用路径均正确。

**Conclusion and Perspective**

在这个脚本中，我们初步实现了根据两个给定蛋白返回两个蛋白间最短作用路径的目标。但是由于时间和编程水平的限制，我们的代码部分还不尽完善。首先是蛋白质互作的路径权重决定问题。如上文所提，我们并没有找到一个提供蛋白质互作效率相关信息的数据库，因此在直接互作的蛋白质路径权重上并没有设置区分，从而导致了结果的不准确。再比如爬取数据的效率较低。通常情况下CPU只有单核在进行工作，对于需要提高效率进行多喝处理的任务，我们可以通过一定的手段让CPU的其他核也进行工作，从而成倍缩短数据处理时间。但是在这里我们使用的是最为简单的方法，即直接在前台使用cmd进行多线程处理，并没有完全实现自动化。以及由于数据库本身信息更新不及时，很多最新的蛋白在数据库里并不能找到完整的信息，甚至出现了一些蛋白的缺失，而这对我们最后结果的呈现有着一定的影响。最后，我们并没能成功实现蛋白质互作网络图的可视化。网络上有一些可以进行数据可视化的软件、插件和相关数据库，如利用Cytoscape软件和StringApp插件联合STRING数据库进行蛋白质互作网络的提取，但我们并没有成功写出能实现这一功能的代码。这些都是我们仍需要提升的地方。

在完成这项project的过程中，我们也学到了许多新的知识。除了对于python编程语言的使用熟练度有了提升以外，我们还学习了selenium的使用和一些其他插件、数据包的安装与使用；学习了正则表达式的使用；初步接触了图论以及完成了BFS的算法实现；在蛋白质数据与编号字典的构建过程以及最短路径的寻找过程中也对数据结构相关的知识有了更加深刻的理解。

最后，我们要特别感谢人工智能方向的丁同学，是他在我们初学python，几乎一窍不通的这段时间里耐心地解答我们的各种莫名其妙的问题，给我们提供了莫大的帮助与支持。是他在我们对于数据爬取一筹莫展的时候给我们提供了使用selenium的建议，也是他在我们磕磕绊绊的代码编写过程中给我们答疑解惑。可以说，没有他，就没有这个基本完成了的project。谢谢你。