

基于串联质谱的蛋白质鉴定 引擎pFind的工程问题

王乐珩

中国科学院 计算技术研究所
前瞻研究实验室 生物信息研究组

2008年11月



中国科学院
计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY

王乐珩

软件工程师

生物信息研究组

计算蛋白质组方向

主要负责pFind引擎的软件架构和工程进度

问题背景

工程架构

优化问题

未来计划

报告内容

- 问题背景
- 工程架构
- 优化问题（发言时间调整，这次来不及讲）
- 未来计划

报告内容

- 问题背景
- 工程架构
- 优化问题（发言时间调整，这次来不及讲）
- 未来计划

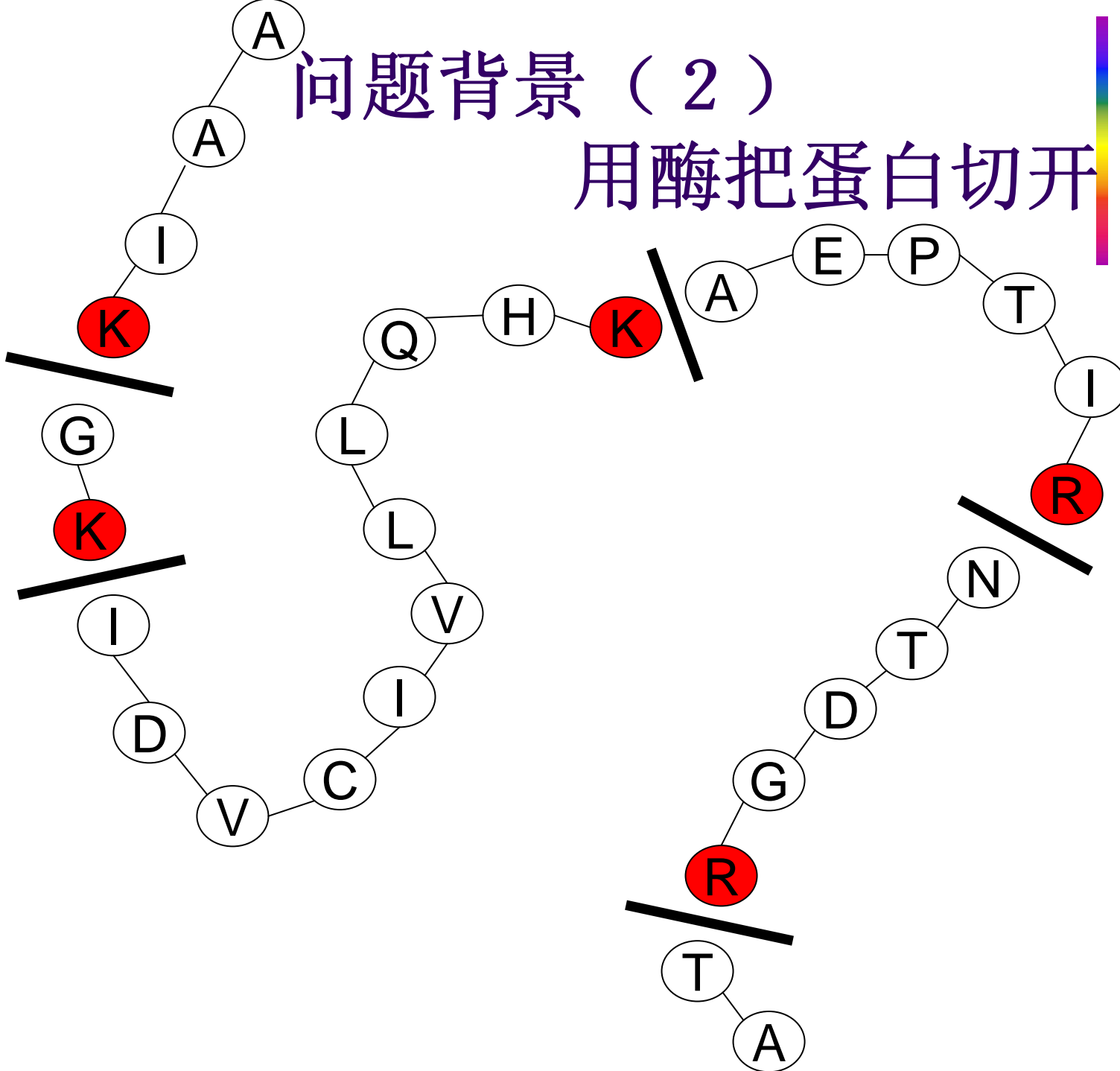
问题背景 (1)

从蛋白质开始



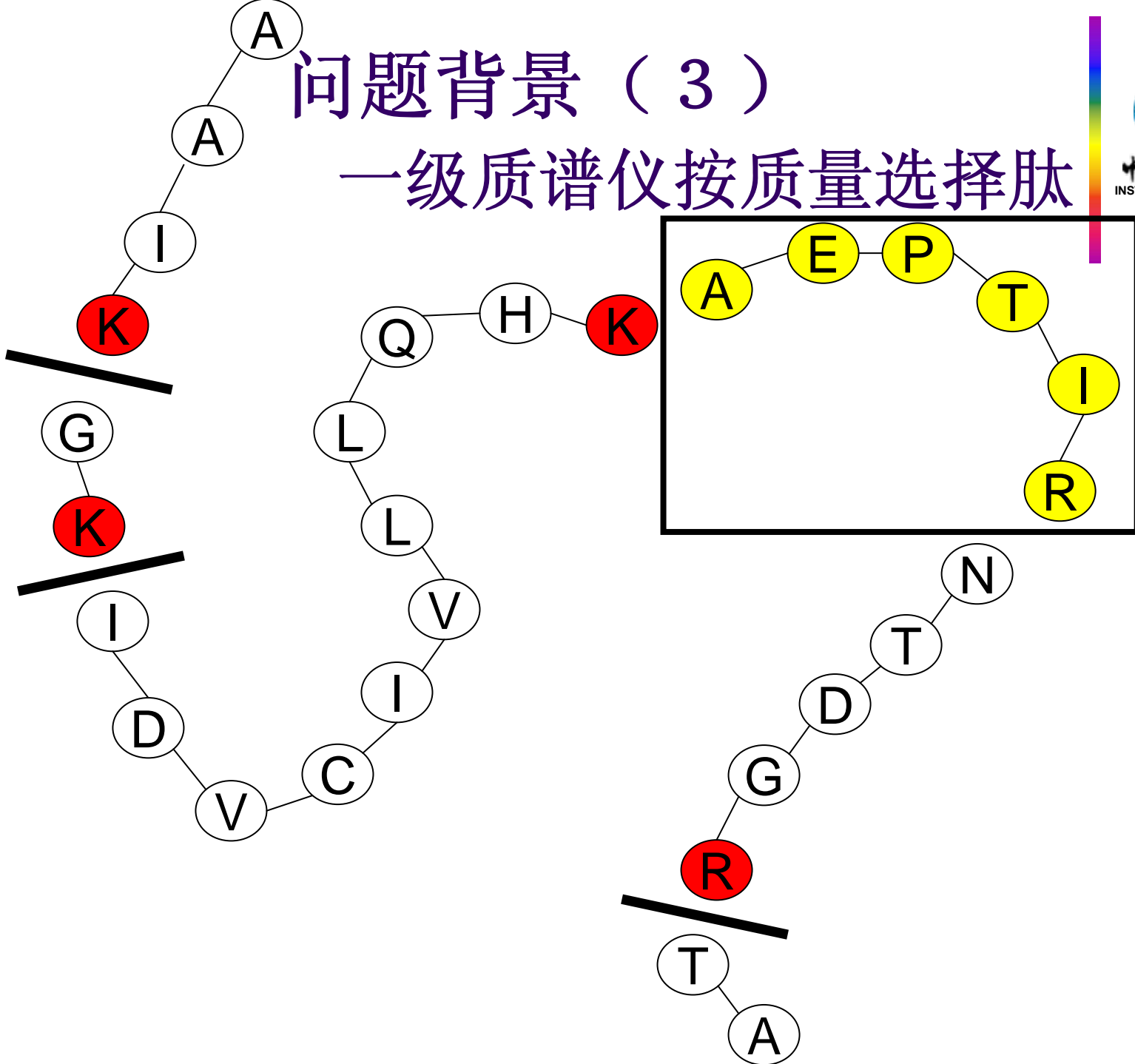
问题背景 (2)

用酶把蛋白切开



问题背景 (3)

一级质谱仪按质量选择肽

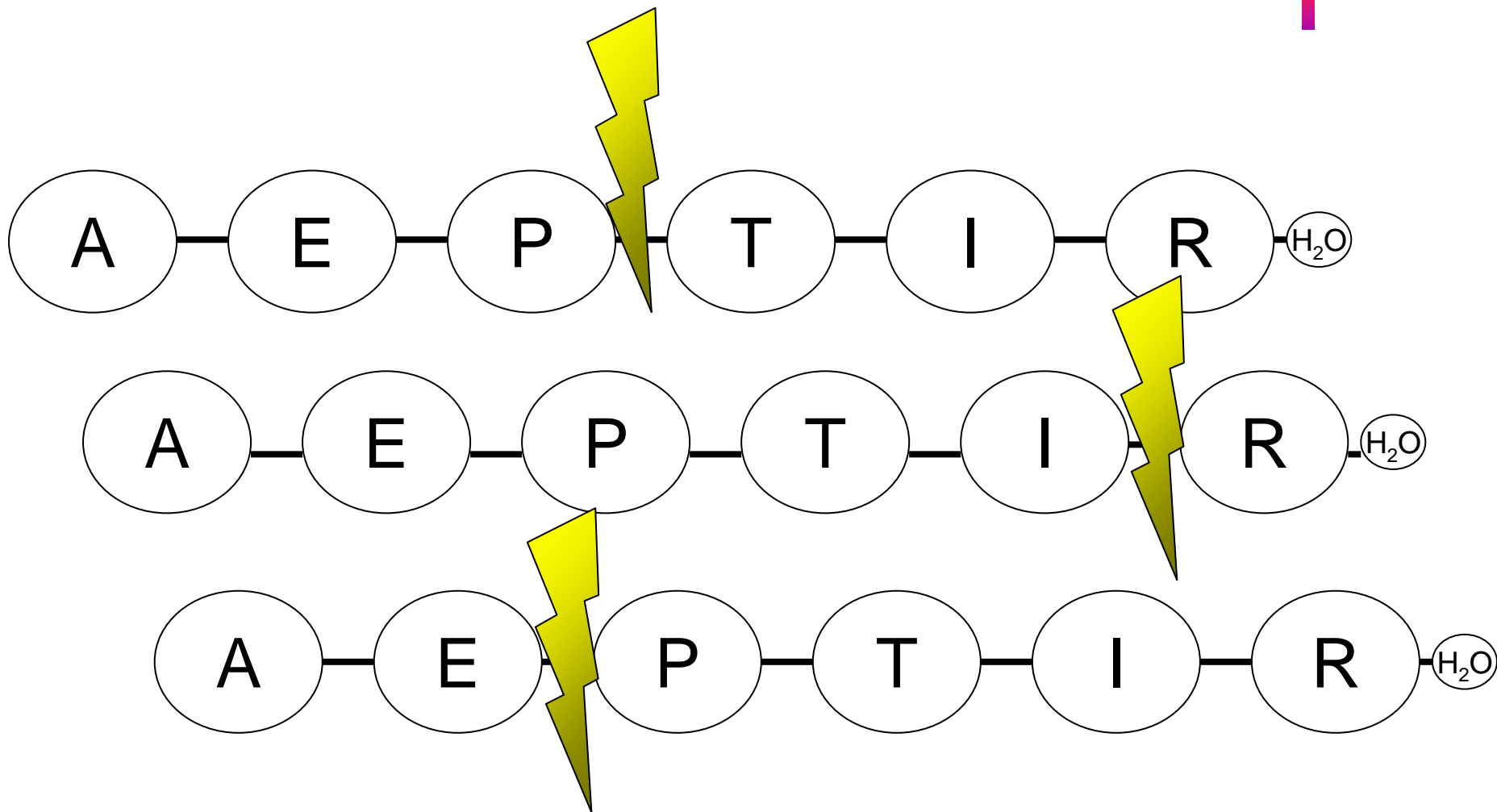


问题背景（4）

二级质谱仪使这个肽碎裂

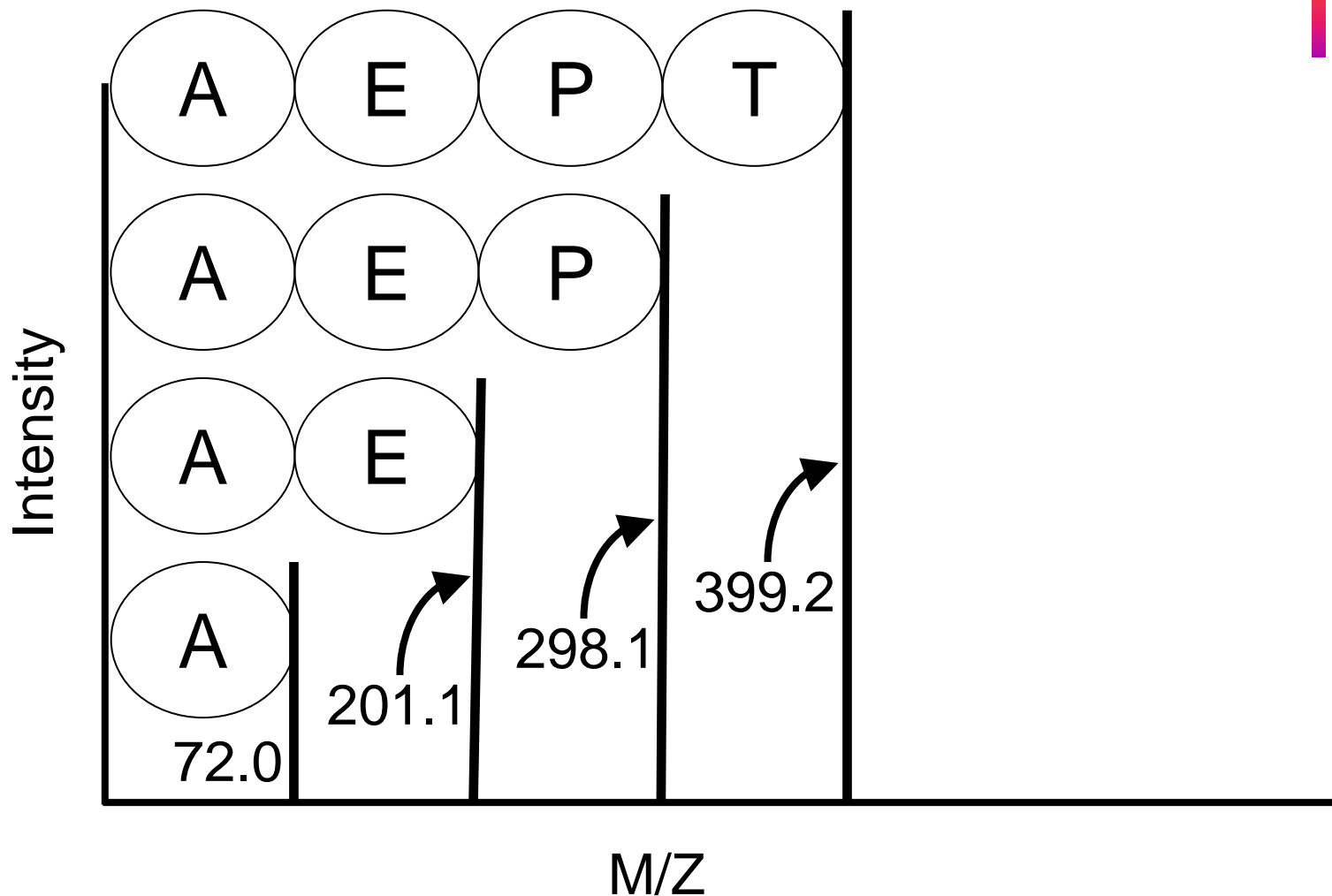


中科院计算所
INSTITUTE OF COMPUTING
TECHNOLOGY



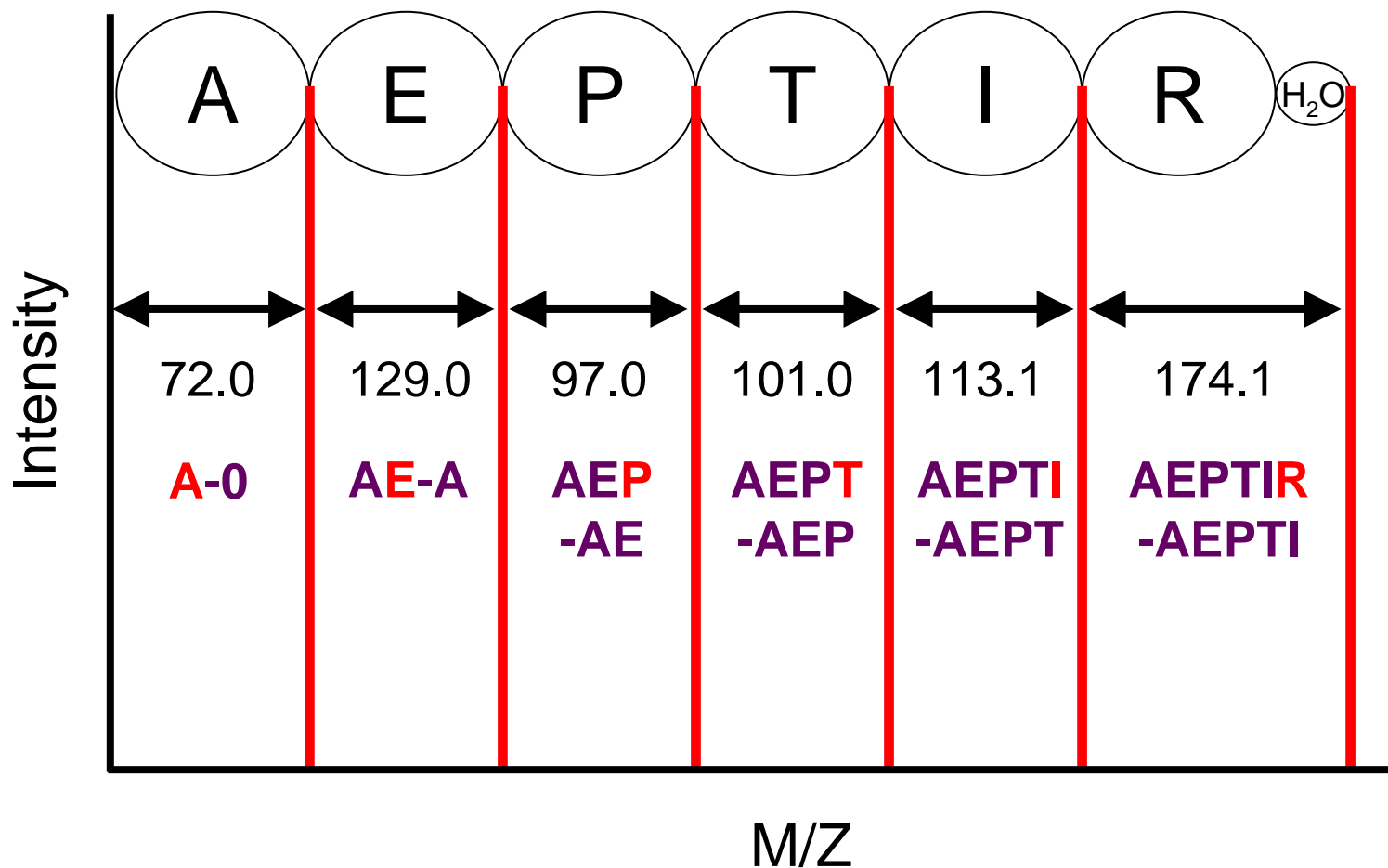
问题背景 (5)

并测量碎片离子, 生成质谱



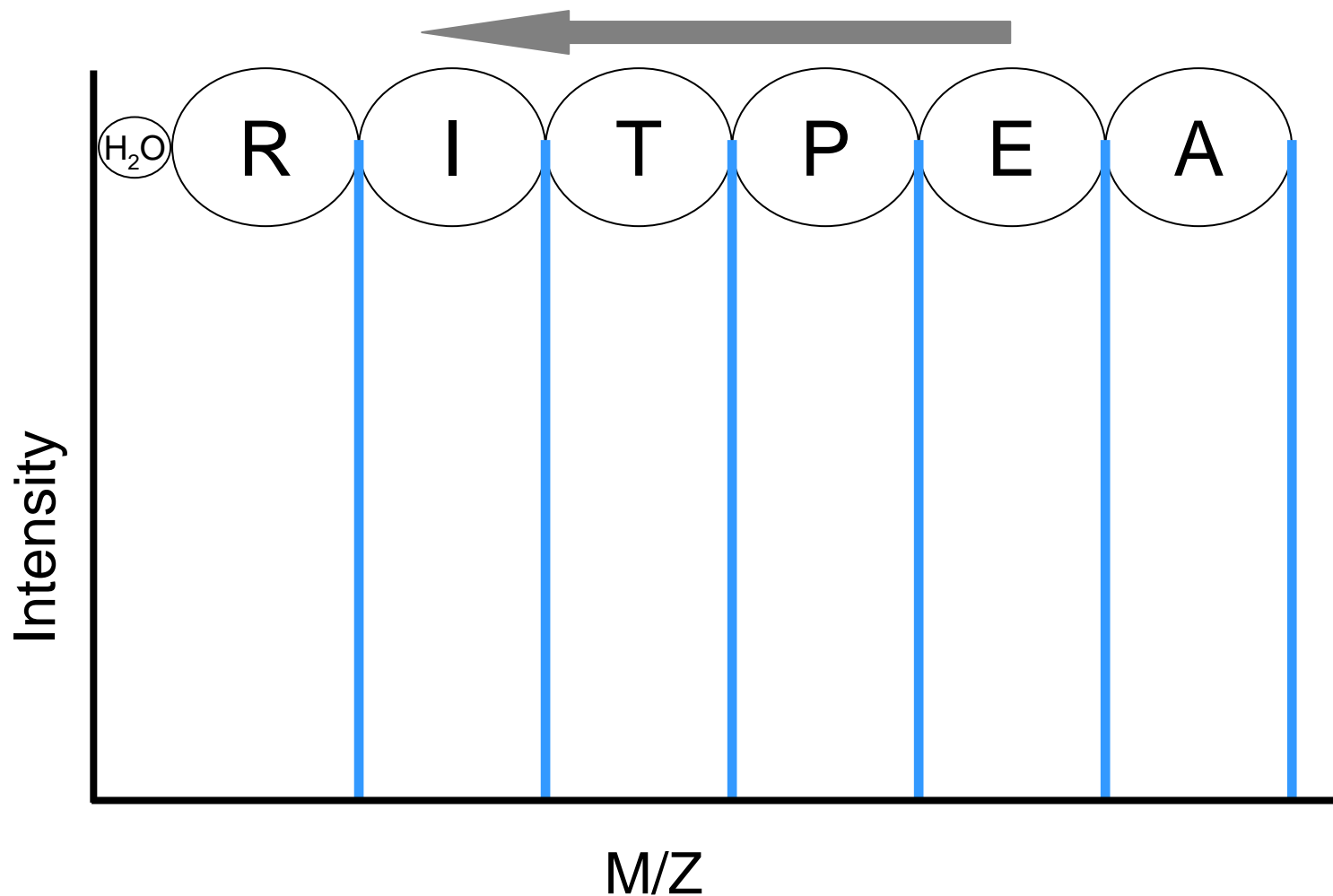
问题背景 (6)

B离子碎片序列

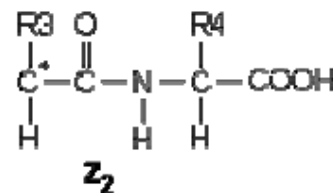
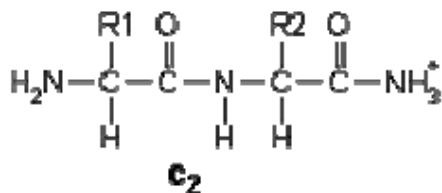
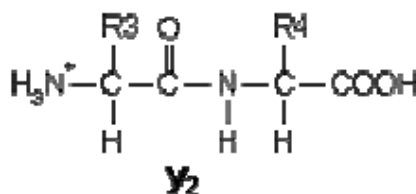
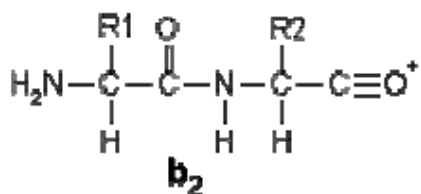
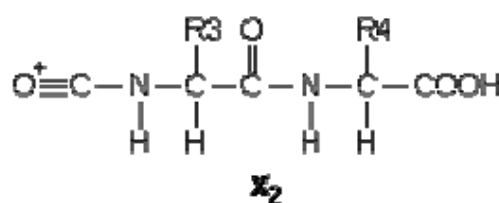
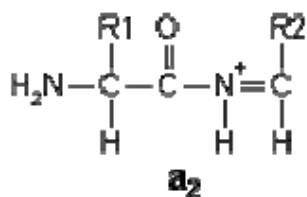
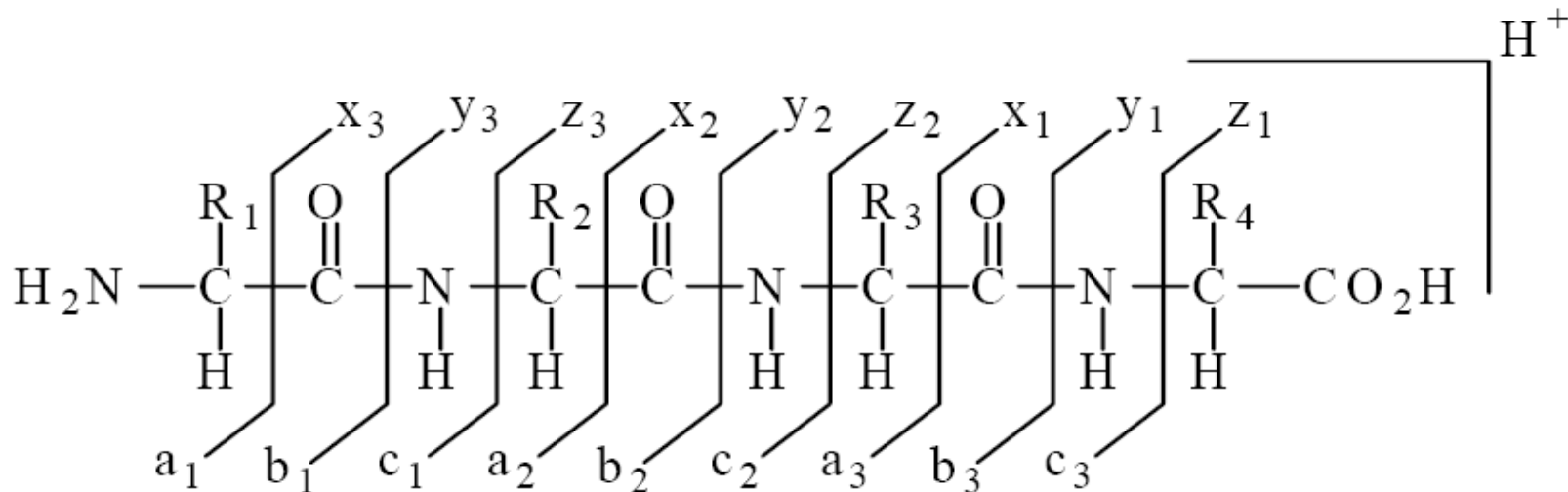


问题背景 (7)

Y离子碎片序列

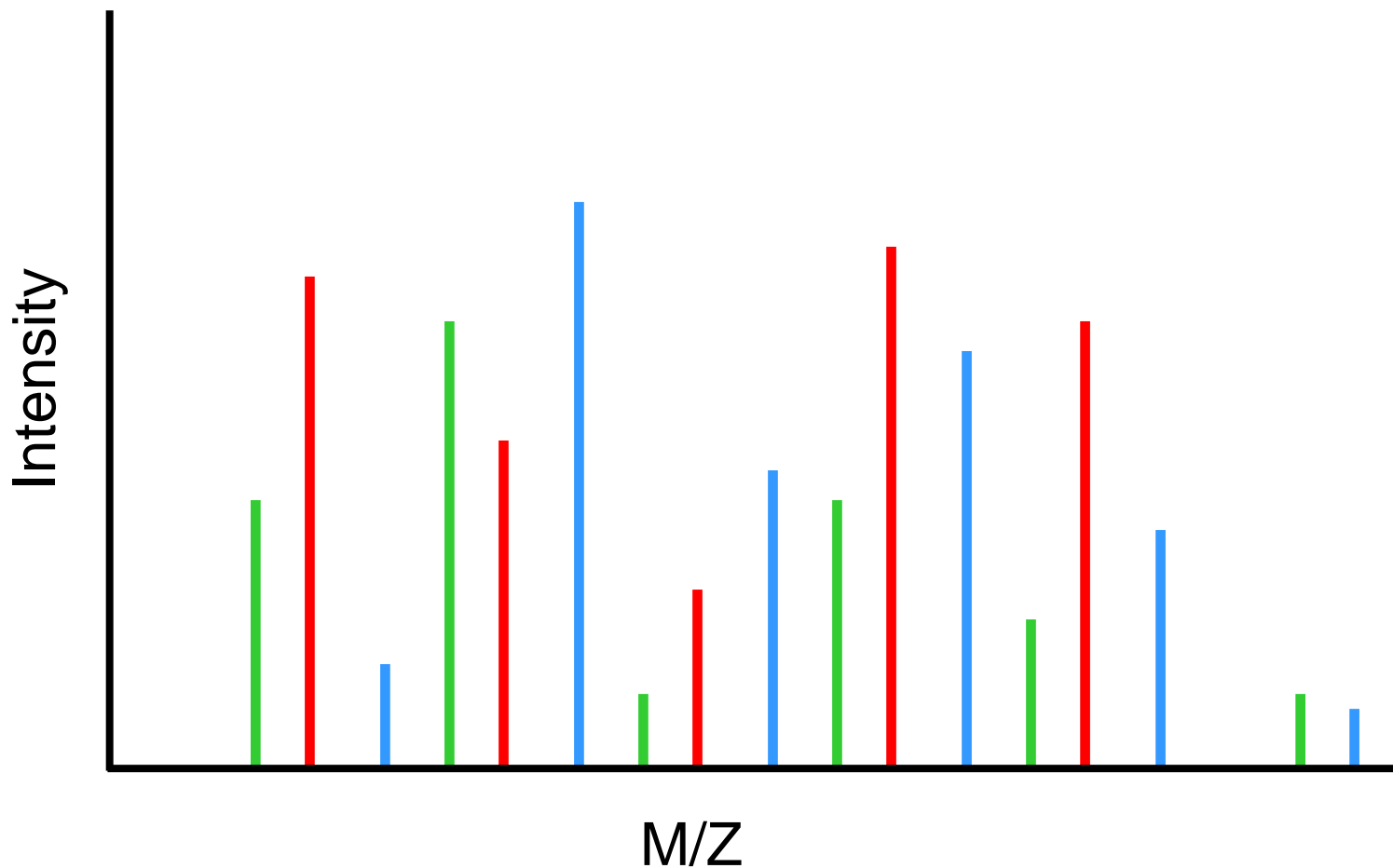


问题背景 (8) 各种离子



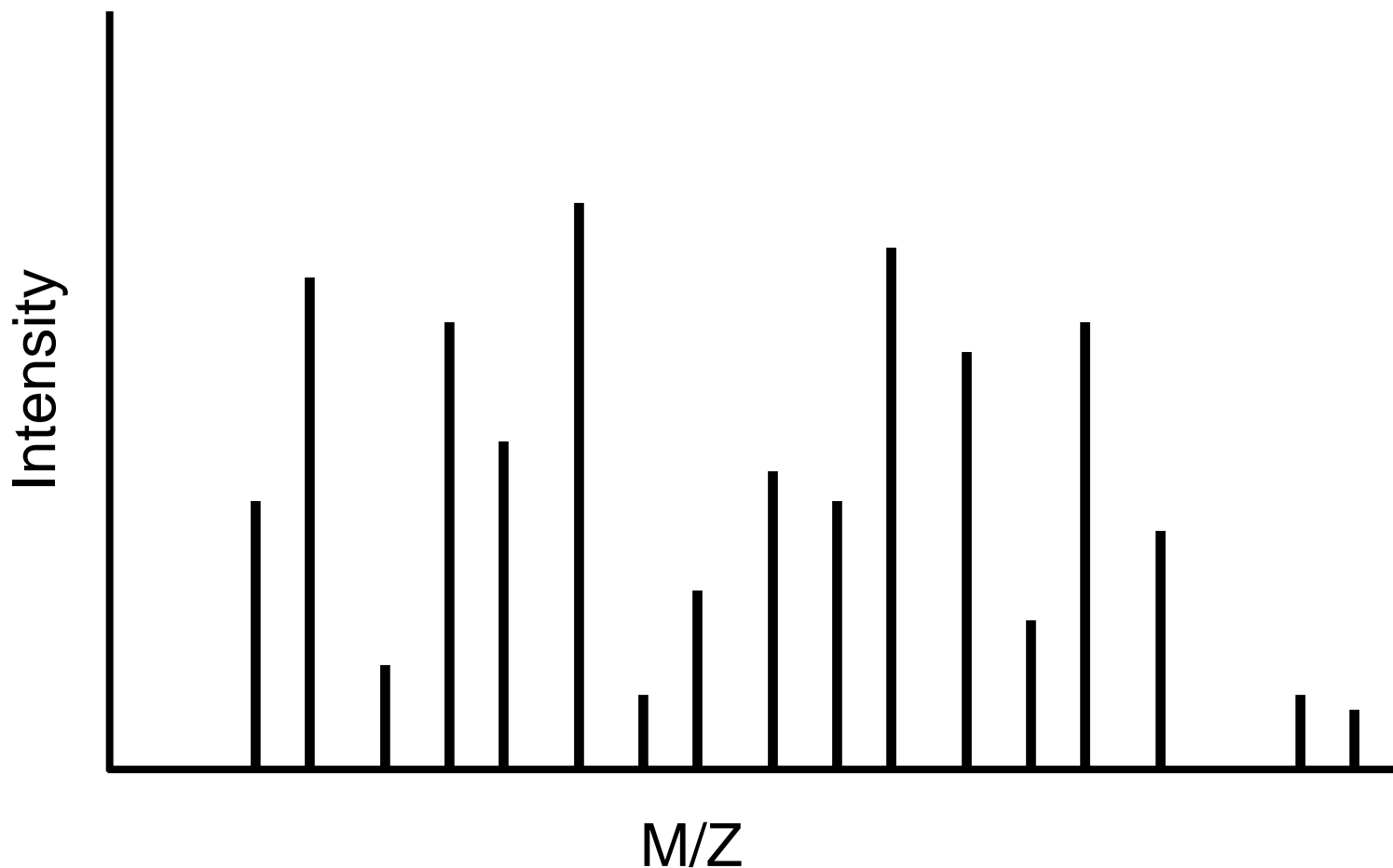
问题背景 (9)

各种离子都混在一起...



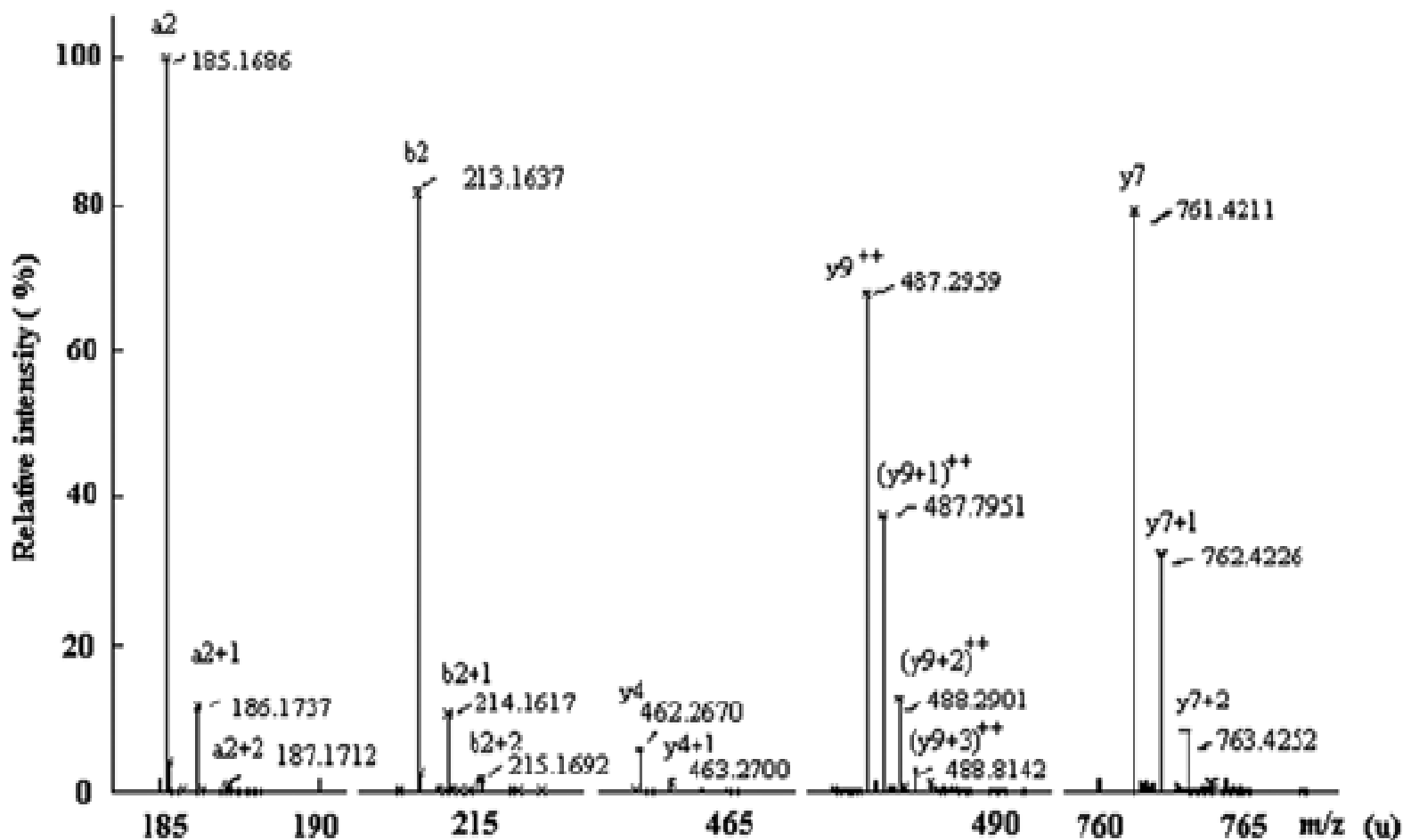
问题背景 (10)

把参考答案蒙上，序列是？

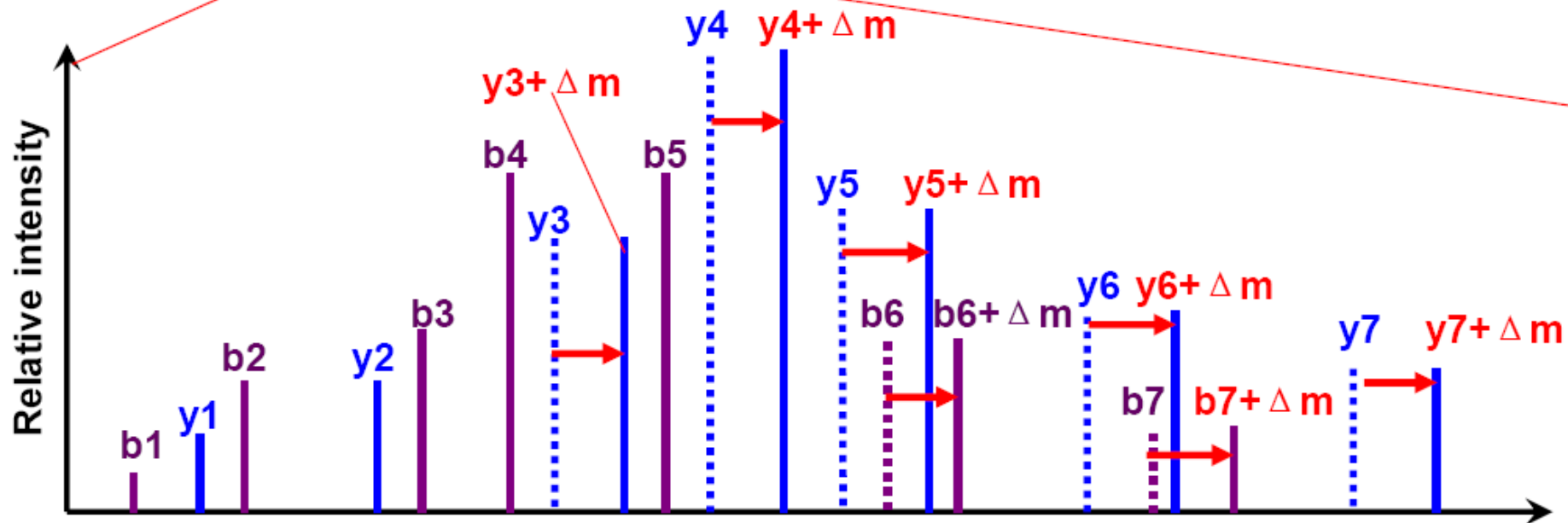
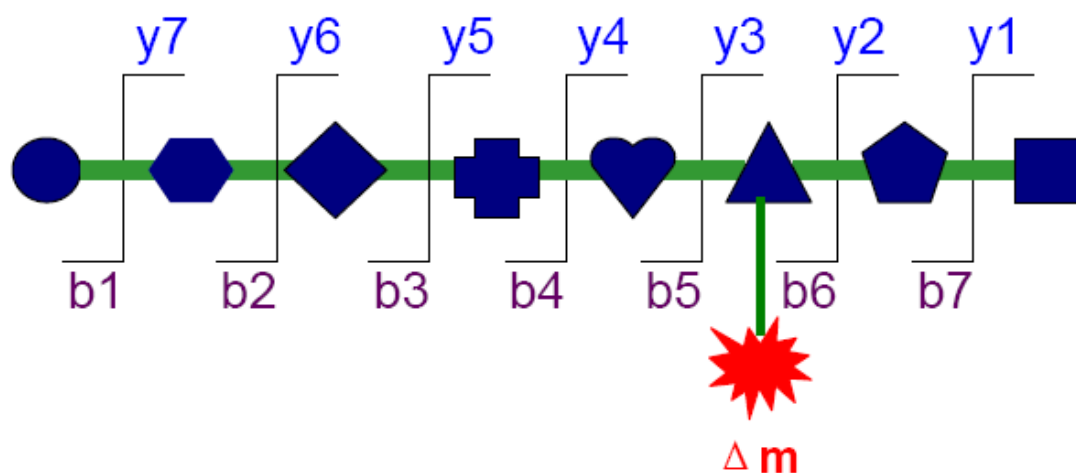
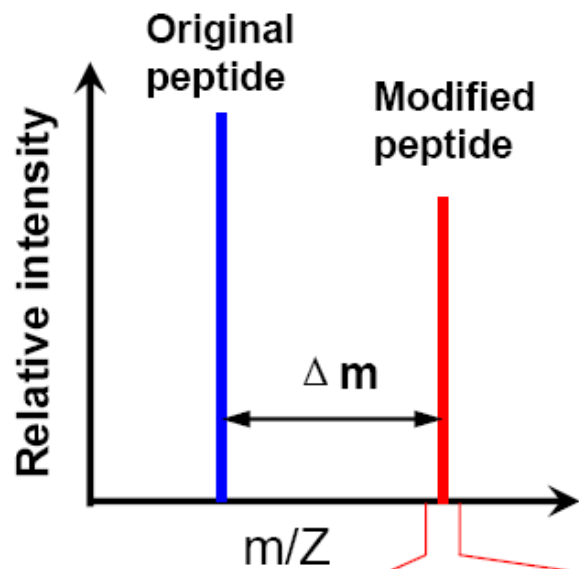


问题背景 (1 1)

还有各种噪音, 例如同位素



问题背景 (1 2) 修饰很重要



问题背景 (1 3) 肽鉴定的规模



IPI(v3.29)Human蛋白库
包含**68,161**条蛋白质

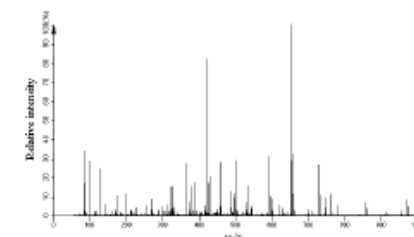
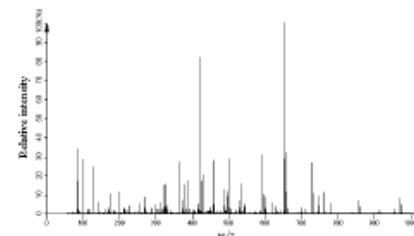
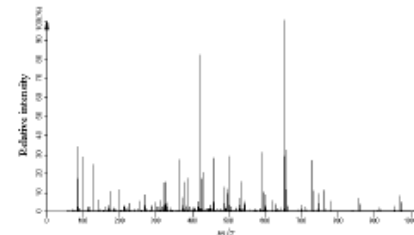
Trypsin酶切, 800到6000Da
间有**3,251,864**条肽段

一次典型实验最少产
生**50,000**张左右谱图

>IPI:IPI00000001.2
MSQVQVQVQNPSAAL
SGSQILNKJQSLLSQPL
MSIPSTTSSLPSENAGR
PIQNSALPSAS...
... ..
>IPI:IPI00419509.2
MAARRGRRDGVAPPP
SGGPGPDPGGGA...SS
CSSSGRSRRCSSSSSSSS
SSSSSSSSSSSSSR...
... ..
>IPI:IPI00845521.1
MQRELVYARGDGPGA
PRPGSTAHPHAIPNSP
PSTPVPHSMP...



MSQVQVQVQNPSAALS
GSQILNK
JQSLLSQPLMSIPSTTSSL
PSENAGR
... ..
MAAR
MAARR
MAARRGR
SSCSSGR
CSSSSSSSSSSSSSSSSSS
SR...
... ..
MQR
ELVYAR
GDGPGAPR
... ..



问题背景 (1 4) 修饰导致组合爆炸



IPI(v3.29)人类蛋白库
包含**68,161**条蛋白质

可能的磷酸化**修饰候**
选肽段数以**百亿**计

一次典型实验最少产
生**50,000**张左右谱图



```
>IPI:IPI00000001.2
MSQVQVQVQNPAAAL
SGSQILNKJQSLLSQPL
MSIPSTTSSLPSENAGR
PIQNSALPSAS...

... ..

>IPI:IPI00419509.2
MAARRGRRDGVAPPP
SGGPGPDPGGGA...SS
CSSSGRSRRCSSSSSSSS
SSSSSSSSSSSR...

... ..

>IPI:IPI00845521.1
MQRELVYARGDPGA
PRPGSTAHPHAIPNSP
PSTPVPHSMP...
```



```
MSQVQVQVQNPAAALS
GSQILNK
MSQVQVQVQNPAAALS
GSQILNK

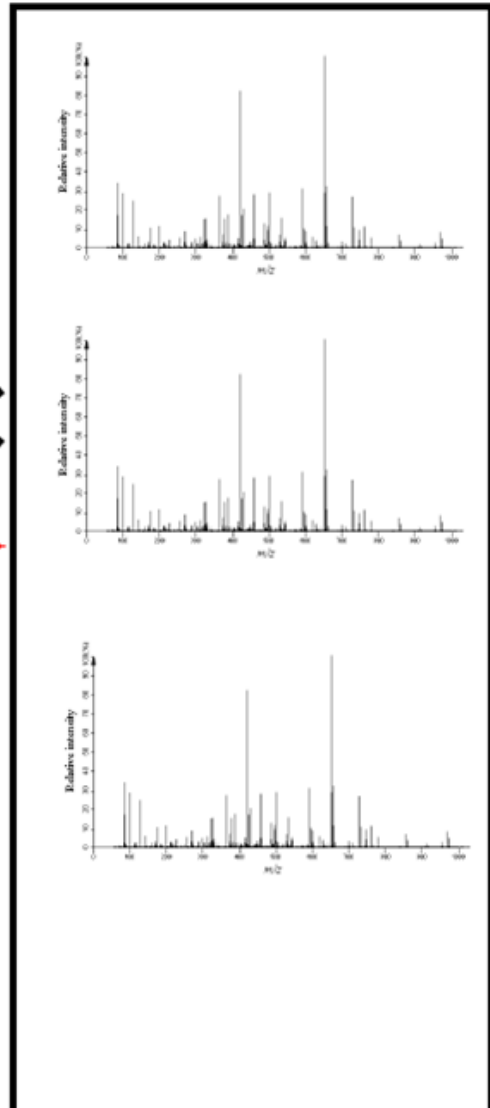
... ..

CSSSSSSSSSSSSSSSSSSSR
CSSSSSSSSSSSSSSSSSSSR
...
CSSSSSSSSSSSSSSSSSSSR
CSSSSSSSSSSSSSSSSSSRC
SSSSSSSSSSSSSSSSSSSR

...
CSSSSSSSSSSSSSSSSSSSR
...
CSSSSSSSSSSSSSSSSSSRC
SSSSSSSSSSSSSSSSSSSR
```



220个



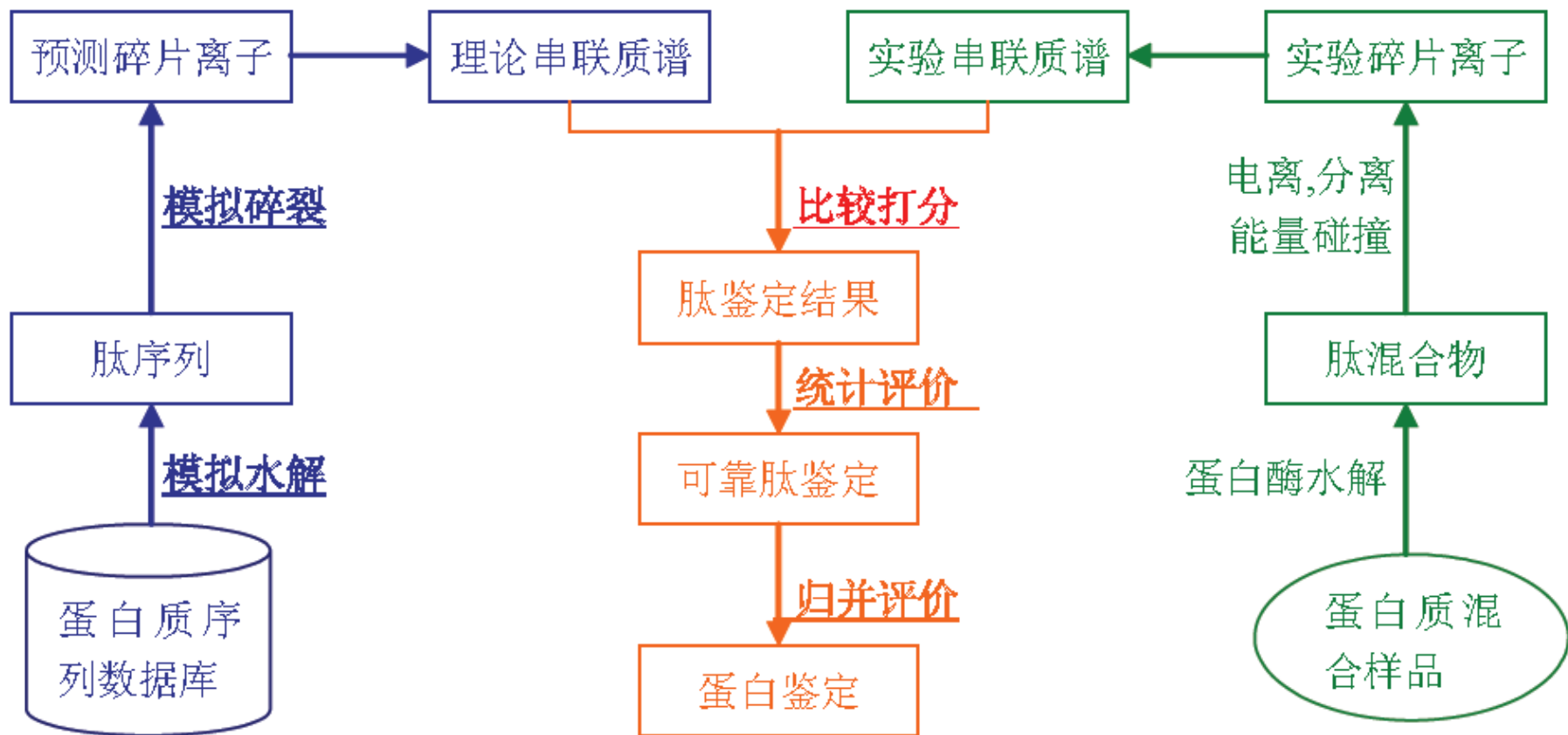
问题背景（15）

一个典型的计算密集型问题

- 08年初，pFind 2.0 Beta 2搜索一个较大数据，计算一周，搜索进度才达到14%
- 各部分多次遇到32位系统进程内存上溢问题。
- 不仅仅关系速度，实际上也涉及到鉴定精度
例如：修饰、半酶切和非特异性酶切
- 计算能力上去了，整个鉴定流程也许就不同了
直接搜索理论组合库？直接搜索基因DNA库？
- 高性能计算技术的挑战和机会

问题背景 (16)

数据库方法



报告内容

- 问题背景
- 工程架构
- 优化问题（时间调整，这次来不及讲）
- 未来计划

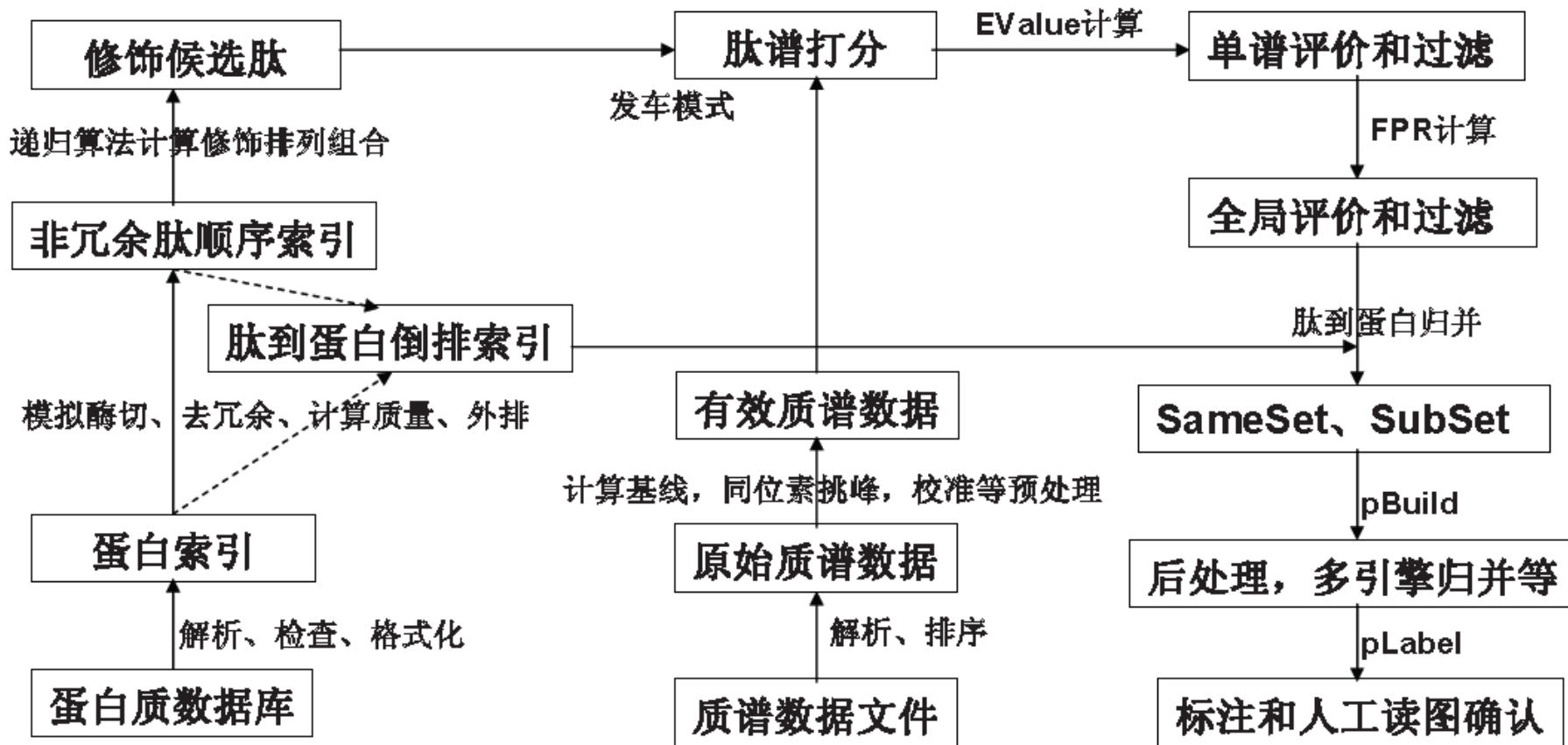
系统架构（1）

OK，前面介绍了领域知识

那么，和前面几位专注于某特定环节算法的博士们不同，作为工程负责人，我面临的工作有什么特点呢？

系统架构 (2)

流程很长: pFind Studio流程



系统架构（3）

升级很快：08年内的版本迭代

- 2008.11.04: pFind 2.2 Alpha 1
- 2008.08.08: pFind 2.1 final release
- 2008.07.01: pFind 2.1 beta 2
- 2008.05.01: pFind 2.1 beta 1
- 2008.02.29: pFind 2.0 final release

系统架构（4）

大型软件：各版本工程量

- pFind 1.0版 62888行
- pFind 1.5版 90147行
- pFind 2.0 Alpha1版（无pCompare） 120607行
- pFind 2.0 final release版 181426行

这里提到的各里程碑之间都有超过2次的彻底重构VSS统计，均有超过85%代码进行了重写。

系统架构（5）

综上，pFind开发有以下特点

- 科研平台，各个模块算法和整体流程一直在快速演进，每3、4个月迭代出一个新版。
- 大型软件，超过10万行代码，历时五年，参与开发人员超过20人。
- 计算密集型应用，空间和时间上都面临挑战。
- 不是in house软件，要在不同的环境下被各种用户安装使用

系统架构（6）

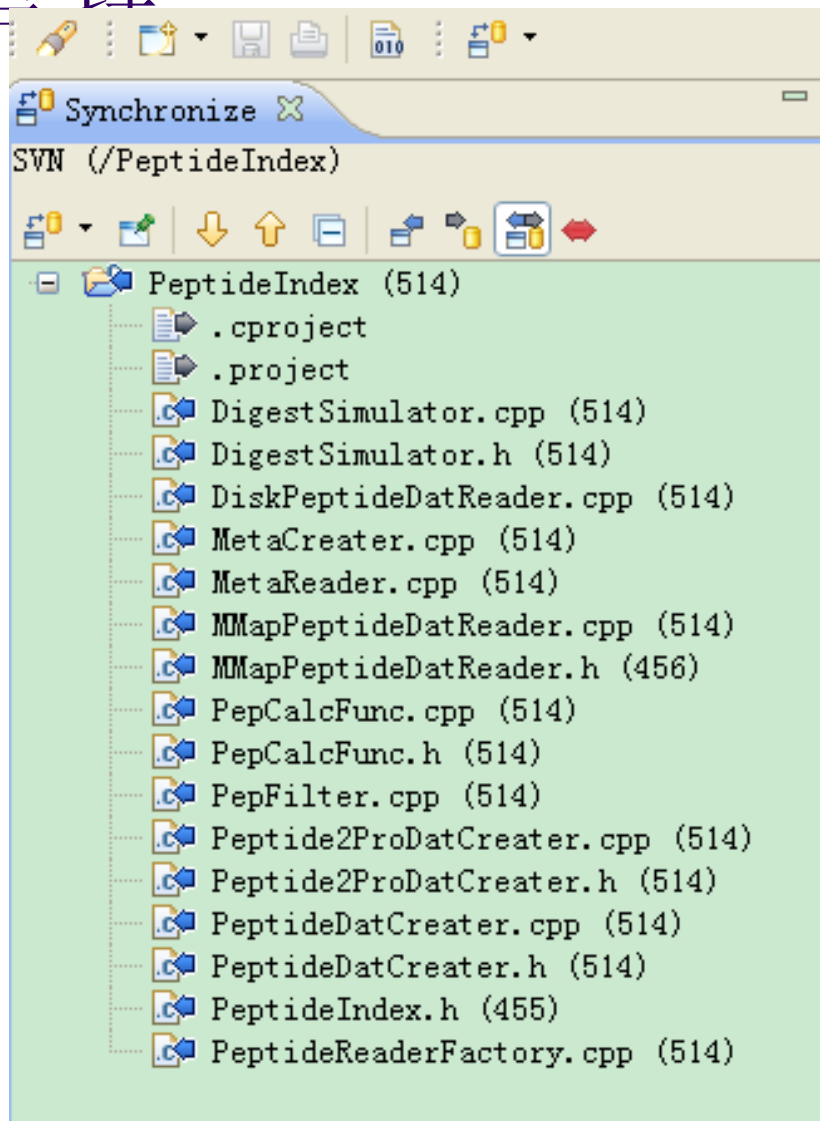
对策一：引入基本保障工具

工欲善其事,必先利其器

- 版本管理：SVN
- BUG管理：BugFree
- 单元测试：CPPUnit
- 性能测试：Oprofile
- 交流备档：Google Docs
- 配置管理：automake
- 网络流量：Google Analytics

系统架构（7）

版本管理



系统架构 (8)

BUG管理



中科院计算所
INSTITUTE OF COMPUTING
TECHNOLOGY

BugFree [Welcome, 王乐珩] [编辑我的信息] [退出] [帮助] [关]

Label [pLabel]

Test Case Test Result **创建 Bug**

查询条件

1 项目名 等于 []
 并且 模块路径 包含 []
 并且 Bug 编号 等于 []

2 创建者 等于 []
 并且 指派给 等于 []
 并且 Bug 标题 包含 []

并且 或者

提交查询 保存查询 重置

结果 1-71/71 100 下一页 自定义显示 全部导出

Bug 编号 ↓	!	Bug 标题	Bug 状态	创建者	指派给	解决者	解决
76	1	将Mascot酶文件转化成enzyme.ini	Closed	王乐珩	Closed	林芳	Fixed
75	1	pBuild还没有异常日志机制	Active	王乐珩	迟浩		
74	1	异常信息日志输出	Active	王乐珩	秀丽蕴		
73	1	取消老的计数器, 只用Google Analytics	Active	王乐珩	王文平		
72	1	pLabel主界面用鼠标滚轮, 快速切换图的时候, 崩溃	Active	王乐珩	秀丽蕴		
71	1	Some meta file in "dbconf.ini" is not a...	Closed	王乐珩	Closed	李由	Fixed
70	1	pBuild里直接调用pLabel, 没有标注结果, 按下一张崩	Closed	王乐珩	Closed	秀丽蕴	Fixed

2.0-Bug 统计报表 - Mozilla Firefox

查看 (V) 历史 (S) 书签 (B) 工具 (T) 帮助 (H)

http://10.29.0.101:8080/bugfree/Report.php?ReportMode=Bug

Bug 统计报表

Bug 项目分布

分布
分布
分布
分布

Active	王乐珩	秀丽蕴		
Closed	王乐珩	Closed	秀丽蕴	Fixed
Closed	王乐珩	Closed	秀丽蕴	Fixed
Closed	王乐珩	Closed	秀丽蕴	Fixed
Closed	王乐珩	Closed	秀丽蕴	Fixed
Closed	王乐珩	Closed	迟浩	Fixed
Resolved	秀丽蕴	秀丽蕴	王乐珩	Fixed
Closed	王文平	Closed	李由	Fixed
Closed	王文平	Closed	李由	Fixed
Closed	王乐珩	Closed	李由	Fixed
Closed	秀丽蕴	Closed	迟浩	Fixed
Closed	秀丽蕴	Closed	迟浩	Fixed
Closed	王文平	Closed	迟浩	Fixed
Closed	王乐珩	Closed	李由	Fixed
Closed	迟浩	Closed	迟浩	Fixed
Closed	秀丽蕴	Closed	王文平	Fixed
Closed	王乐珩	Closed	迟浩	Fixed

系统架构（9）

风险管理

Google Docs
pFind 2.1 Alpha 1项目手册 saved on July 5, 2009 12:16 AM by ...

File Edit View Insert Format Table Tools Help

五、风险识别和对策:

Risk: 索引部分开发强度很大,在陌生的gcc+Eclipse+cdt环境下编码,有可能出现进度拖延和设计质量不达标。
Solution:

- 1.采用严格的双人编程和代码审阅,尽可能提高编程效率。(2009.4.13)
- 2.在...完成负责模块的情况下,支援索引模块。(2009.4.13)
- 3.尽早进行索引模块的单元测试和大规模测试。(2009.4.13)
- 4.明确重点,力保蛋白索引和非冗余肽索引部分,在最悲观情况下放弃倒排索引部分。(2009.4.13)

Follow up:

4月22日:目前看,代码质量得到了保证,设计模式充分得到应用,但是进度上有所拖延。
4月13日:经过集体code review,对代码提出了进一步的意见。
4月12日:截至目前,索引部分已经按照设计模式进行了架构重构。
4月8日:经过code review发现代码有include混用问题,必须进行一次代码重构,预防BUG。

Risk: 搜索引擎部分,接口和结构设计部分没有专门的“质量看门狗”,导致设计上的妥协和后期的被动重构。
Solution:

- 1.明确质量底线,“天条不准越过”,工程要求制度化,文档化。(2009.4.13)
- 2.code review和双人编程制度化。(2009.4.13)
- 3.我本人要负起责任,不再那么好说话,磨利爪牙,准备为了捍卫原则而争吵。(2009.4.13)
- 4.了解“...”的进度,如果那边进度超前完成的话,再把“...”请回来。(2009.4.13)
- 5.使用BugFree把改进意见收集起来,白纸黑字,准确日期,作为“呈堂证供”。(2009.4.13)

Follow up:

4月22日:接口和结构设计部分基本已经完成,接下来的任务是形成技术报告。
4月13日:已经经过了两次集体code review,形成代码审核和双人变成制度,但BugFree还没有用起来。
4月8日:目前BugFree还没有实际用起来,肽索引部分还没有双人编程和重构。

Risk: 环境干扰,或项目管理不力,导致士气不足,资源不足,开发效率下降。
Solution:

- 1.提前公布进度计划,保证这个计划不被打断。(2009.4.13)
- 2.我本人要负起责任,为项目争取一切有利资源。(2009.4.13)
- 3.不加班原则:提高士气的最好途径,就是保证团队8小时高效工作,每周提前达到里程碑,准时放假。(2009.4.13)
- 4.减少一起不必要会议,文档,我本人负责一切“打杂”工作,包括开发文档撰写。(2009.4.13)

系统架构 (1 0)

Joel 12条

1. 使用版本控制工具吗? ✓
2. 有自动Build机制吗完成连编吗? ✓
3. 每日自动集成吗? ?
4. 有BUG管理工具吗? ✓
5. 任何事之前, 优先修复BUG吗? ✓
6. 依据Todo List和milestone吗? ✓
7. 对用户需求和交流进行归档吗? ✓ (Google Docs)
8. 程序员拥有安静的工作环境吗? ?
9. 用到了资金能力内最好工具吗? ✓
10. 有专职QA吗? ?
11. 新聘人员试用期编码吗? ✓
12. 进行走廊易用性测试吗? ✓

4年前我加入生物信息组时为4分, 目前我们接近10分。感谢领导的支持

系统架构（1 1）

对策二：引入模块化观点

- 把从系统内核里逐步剥离出众多单一模块。
- 引入设计模式
- 让所有团队成员都理解基础的软件架构原则，例如：“接口上移，功能外抽”，“高内聚，低耦合”等
- 不断发起重构和**code review**，在原则问题上不让步。

系统架构（1 2）

关于设计模式

设计模式（**DP**）：面向对象设计中的经典套路，类似武侠里的招数。最初由建筑学引入软件工程。

设计模式是共性的设计思路，最好有万行以上代码经验。没有太多实践经验读**Gof**的《设计模式》可能有困难。

系统架构（13）

pFind Studio代码中常用的模式：



- 工厂方法模式（“第一模式”，虚函数的基本应用，pFind模块机制的基石）
- 单件模式（大多数配置和资源限制型类最好使用，例如索引的元数据和内存映射等等）
- 文档视图模式（MFC库强制的模式，如果不用，就没办法用MFC体系，pBuild和pLabel都以该模式为基础）
- 外覆模式（重构时经常使用，ACE库的所有的方法都是用这种方法和操作系统API解耦）
- 门户模式（SearchEngine.dll。该模式向上层提供服务，输入选择参数，返回结果，隐藏内部架构和流程）
- 模板模式（事件回调机制，依赖注入机制）

系统架构（14）

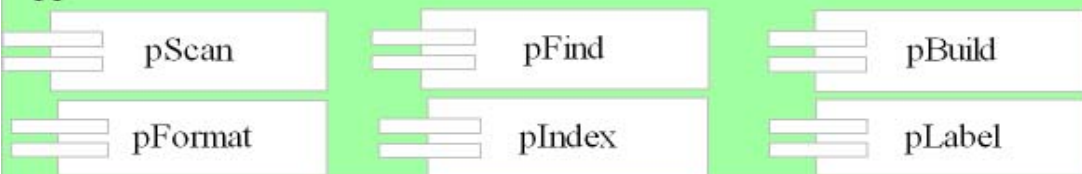
反模式，更常见说法是**bad smell**



维基百科上列出的“臭味”，见到它们就离发生BUG不远了。

- **Input kludge**: 没有防御代码处理非法输入和参数异常
- **Blind faith**: 不检查bugfix或子函数返回值
- **Parallel protectionism**: 相似的功能使用克隆体来处理
- **BaseBean**: 继承一个工具类，而不是内含调用它
- **God object**: 在某个类集中了过多功能
- **Object orgy**: 无所不在的对象，全世界都在不受限制地访问它的各种内部细节
- **Poltergeists**: 某对象唯一作用是把信息传给其它对象
- **Dependency hell**: 版本依赖混乱，常见的DLL hell

Applications



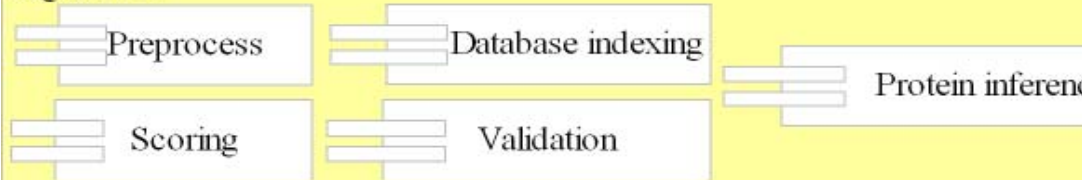
Search Engine



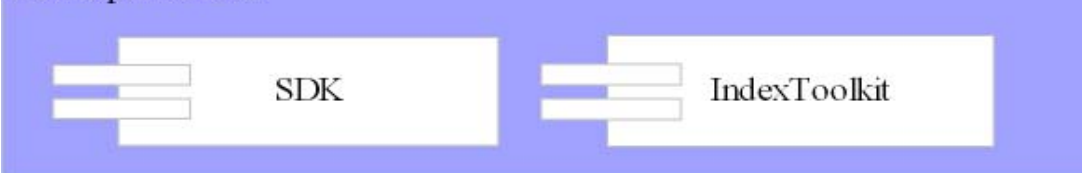
Tools



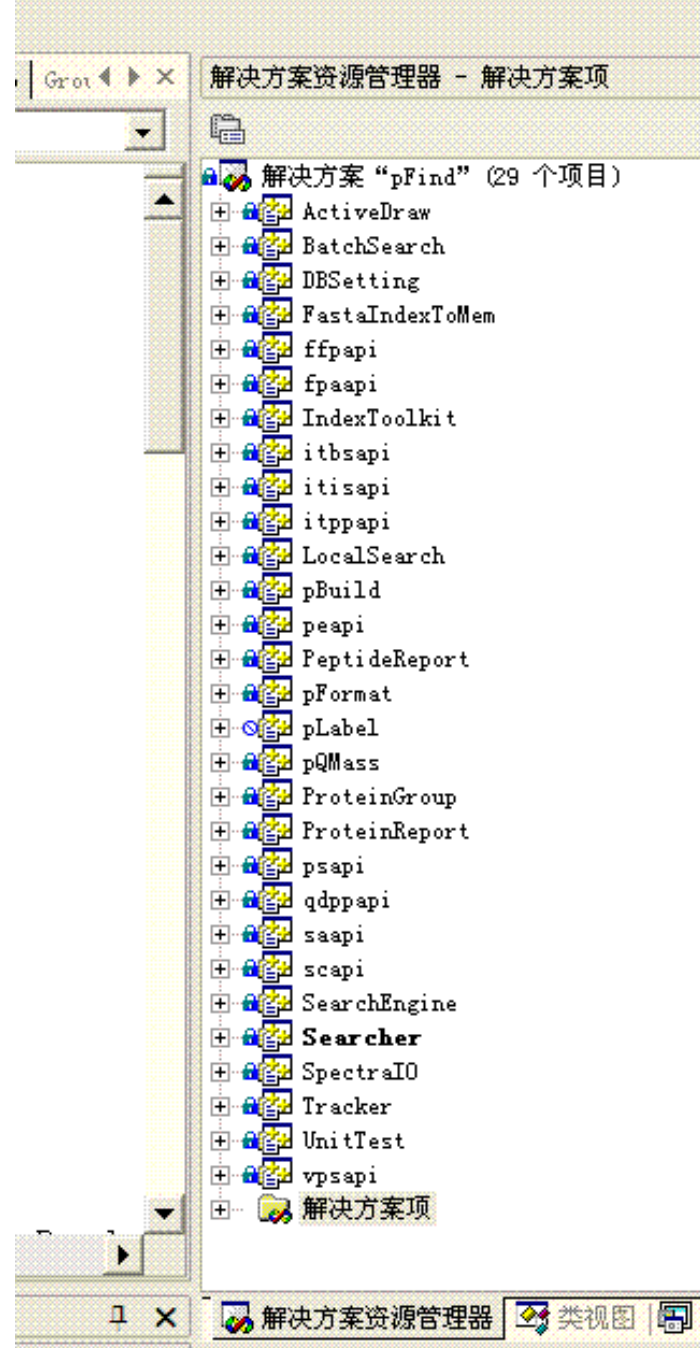
Algorithms



Development Kits



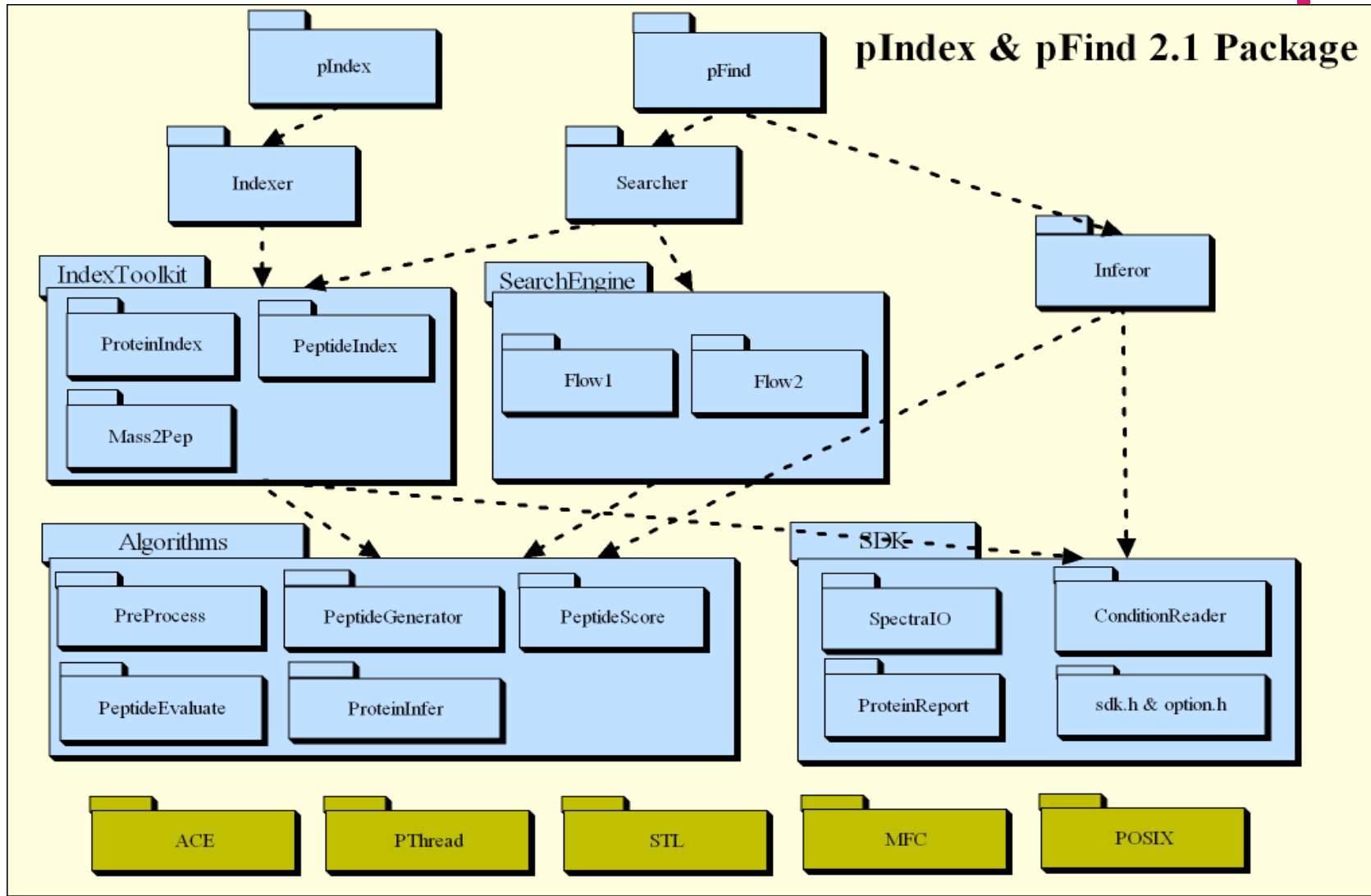
Platforms



38个模块，20万行代码

迭代开发和重构 (15)

UML Package Diagram



系统架构（16）

对策三：引入XP软件工程策略

经典软件工程：一切都可以在设计阶段预计好，开发阶段是工人照图纸加工安装而已。因此设计和实现是重点，测试和重构是补充；

敏捷软件工程：作为思考创作产物，软件做出来之前，你不可能确切预计100%情况，不仅实现细节很难完整预知，用户需求也只能在开发反馈中挖掘。因此最重要的是，时刻保证模块的可拓展性、鲁棒性和可维护性，没有bad smell。以便迎接随时可能发生的航向变化。因此测试和重构才是重点。

系统架构（17）

重构和迭代开发

重构：不影响对外功能，整理代码内部，以增强可维护性和性能。通俗来说，重构就是从“反模式”到“设计模式”，打扫内务。

XP要求，被称为重构必须有一个硬性前提，就是时刻有单元测试作为“脚手架”，否则称为“推倒重写”。典型重构不能超过2小时。

系统架构（18）

重构和迭代开发

迭代开发：小版本快速演化，每个版本只关注有限的几个**features**改进，快速推出原型，展开**Beta**测试，一般每个版本不超过3个月。迭代开发是软件工程发展**40**年来，被证明**唯一可行**的软件开发方式。

pFind组内经验表明，系统模块没有经历两次以上的重构，是不可能成熟的。

系统架构（19）

重构和迭代开发

为什么是两次以上？原型阶段：针对特例；第一次迭代：考虑推广和复用；第二次迭代：验证和调整。

迭代开发的成功的一个心理学原因在于，短期的，可见的里程碑，对团队士气有帮助。

有没有注意到：微软的产品都是在**3.0**以后天下无敌；而**Google**推出新服务的速度很快，但推出后都会有两年以上的**Beta**重构阶段。

系统架构（20）

重构和迭代开发

从pFind 2.1开始，已经迈上了三个月一次升级的良好节奏：

第一月推出Alpha，后两个月通过大量Beta测试完善产品，修正BUG，重构山寨代码。

这种固定节奏被证明非常高效。很多常见软件，例如Eclipse、Firefox、Mascot、都是以三个月或半年为周期进行迭代。

系统架构（2 1）

双人编程

双人编程指在软件项目中由两个程序员同时编写同一段代码的软件工程方式。两人在同一台机器上工作，其中一人操作键盘的同时，另一个在旁边**review** 编写出的代码。操作者从战术上关心当前编写的每一行代码，观察者确认语法规范，并从战略上考虑整个程序。他们频繁地交换彼此角色。

系统架构（2 2）

双人编程

- 双人编程增强勇气：两个人交流，程序员们会倾向于尝试和评估更多技巧和方案；
- 双人编程促进团队合作：模块不是单人写的，代码是团队财产，不属于某个开发者的；
- 双人编程有利于知识的传播：越多的开发者互相配对编码，关于系统的整体知识越能传播到整个团队；
- 双人编程提高生产力：单个程序员精力爆发后，必然面临周期性的思路呆滞（**inactivity**）。双人可以协调步调，当其中一个劳累时，他们可以调换角色，整体工作会始终保持一定强度，而这种强度不是单人可以吃得消的；
- 双人编程提高士气

当然如果是一男一女结对工作，效率就更高了☺

系统架构（23）

对策四：最关键的因素是“人”

现代软件工程对工程师的要求提高了：

- 一开始就关注细节，关注**Bad Smell**和**BUG**，而不是先告诉拼凑胡子工程，指望以后再打扫战场。（这也是瀑布模型的弱点）
- 强烈关注用户交流，整个开发过程中不断交流。有热情的心态和专业的眼光。
- 把自己当提供解决方案的专业人士，而不是黑屋子里不见人的工匠。

系统架构（2 4）

人的培训：架构设计

在设计阶段，带着投影仪到会议室去，少做哲学上的讨论和方法论的思考，直接关注细节，对某个模块的接口进行充分精致的完善，这种设计是直接诉诸于代码的。也就是直接在屏幕上敲出有关的代码，编译，根据大家的讨论进行迭代。

系统架构（25）

人的培训：架构设计

这样做的好处：

- 接口设计迅速落在实际代码
- 团队充分交流，理解架构师的设计思想
- 设计方案通过**code review**进行补充和完善。

pFind 2.1的架构设计，在后期几乎没有变化，实现中工程原则没有打折扣。

系统架构（26）

人的培训：架构设计

架构师一开始就应该对整个架构有明确思路。他参加这个会是为了：

- 一场答辩，通过当场编码和别人对代码的讨论和审核，验证设计的合理性；
- 对程序员进行培训；
- 吸收意见进行完善和迭代‘

系统架构（27）

人的培训：代码审核

代码审核和编程风格标准是最基本的工程要求

- 和Google公司一样，没有两个人会签，不准check in入主版本。
- 用throw异常代替return false。（一个小故事）
- Google公开了他们公司内部的C++编程规范，组里年轻人找来一看，和我们的check list几乎完全重合。

系统架构（28）

人的培训：Joel on software说



当你处在不那么professional的环境，又没有权力影响别人，怎么办？

你可以装个CVS自己先用；你可以装个Bugzilla自己先用；你可以列出自己的Todo List，抄送给所有相关人；你可以拿着BUG满世界追，直到负责这个模块的家伙修正它；

你可以不断在别人耳边唠叨这一切有多重要，直到他们无奈地同意试试；最后，你可以在那些不愿负责任，甚至以此为荣的蠢蛋面前坦率地保持自尊和冷静。

请相信，你能成为最棒的软件工程师。

系统架构（29）

走廊测试和用户现场Beta测试

开发完毕后，首先进行走廊用户（未参与开发，没有实际操作以往版本经验，不依赖用户手册和人工指导的用户）进行试用，并给出改进意见。

到用户现场进行Beta现场使用，收集用户反馈，观察用户操作，这是我们组开发工作中，仅次于迭代开发的原则。需要有好的心态和专业素养。（Joel说，只做in house software的程序员没出息）

报告内容

- 问题背景
- 工程架构
- 优化问题（发言时间调整，这次来不及讲）
- 未来计划

未来计划 (1)

- ~~4月29日，pFind 2.1跑通Cygi数据 (Alpha1)~~
- ~~5月30日，大规模全面测试，调试优化 (Beta1)~~
- ~~7月1日，在至少2个合作单位现场试用 (Beta2)~~
- ~~8月8日，正式发布pFind 2.1 (Final Release)~~
- ~~11月4日，pFind 2.2跑通 (Alpha1)~~
- 年底前，在至少2个合作单位现场试用 (Beta2)

未来计划 (2)

开发出色的软件，把Mascot打得满地找牙！



谢谢！