HTML5设计网页动态条幅广告(Banner) 已经加上完整源代码

2016年5月6日　　13:29

横幅广告(Banner)：

1.横幅广告是网络广告的常见形式，一般位于网页的醒目位置上；当用户单击这些横幅广告时，通常可以链接到相应的广告页面；

2.设计横幅广告时，要力求简单明了，能够体现出主要的中心主旨，鲜明、形象地表达出最主要的广告意图；

3.横幅广告可以使静态图像，也可以是动态图像。一般而言，与静态横幅广告相比，动态横幅广告更醒目，更能吸引观众的注意力；

4.当然这还是在恰当适合的前提下（讨厌那种弹窗式和悬浮式的广告）使用不当会造成意想不到的后果，甚至因此观看者的反感造成恶性循环，从而对广告原本的作用大打折扣；

5.设计横幅广告时，究竟是采取静态形式还是动态形式，取决于哪种形式能够把要表达的信息准确、快速地传递给观看者。

设计过程：

（一）编写HTML5代码

1.输入单击Banner时要链接的网站

```
<a class="banner" href="http://yamoo9.com">
```

2.向Banner中添加图片和文字并使用class属性标识元素

```
    <a class="banner" href="http://yamoo9.com">
        <img class="banner-logo" src="images/banner-logo.png" alt="yamoo9.com"
width="167" height="134" />
        <p class="banner-desc">开放知识讲座, 视频Cast!<br /> 分享设计心得的乐园!<br />
        <strong>- Yamoo9</strong></p>
    </a>
```

（二）编写CSS3样式表

1.控制body样式

```
body {
    padding: 20px;
```

```
    background: #333;
}
```

2.控制Banner样式

```
a.banner {
    display: block;
    width: 728px;
    height: 176px;
    border: 1px solid #555;
}
```

3.设置横幅广告的Logo标志

```
.modern .banner-logo {
    position: absolute;
    top: 20px;
    left: 270px;
}
```

4.向Banner上的文字应用字体

```
.modern .banner-desc {
    font: 32px/1.1 "NanumPenWeb", "方正静蕾简体", "Nanum Pen Script";
}
```

同时还需要在HTML5代码中添加Web字体服务

```
<title>CSS3 Banner Design - 动画Banner设计</title>
<link href='http://api.mobilis.co.kr/webfonts/css/?fontface=NanumPenWeb'
rel='stylesheet' />
```

5.设置Banner字体的位置与颜色

```
.modern .banner-desc {
    opacity: 0;
    position: absolute;
    top: 39px;
    left: 170px;
    font: 39px/1.1 "NanumPenWeb", "方正静蕾简体", "Nanum Pen Script";
    color: #4ec1cd;
}
```

```
.modern .banner-desc strong {
    font-size: 23px;
}
```

6.设置鼠标指针未移动到Banner上的Banner

```
a.banner {
    position: relative;
```

```
    background:
        url(../images/banner-arrow.png) no-repeat -100px 140px,
        url(../images/banner-photo.png) no-repeat -40px 220px,
        url(../images/banner-09.png) no-repeat -20px -380px,
        url(../images/banner-bg.png) no-repeat 0 0;
}
.modern a.banner:hover, a.banner:focus {
    background-position:
        20px 140px,
        -40px 20px,
        -20px -90px,
        0 0;
}
```

使用transition函数完成一系列的图片移动操作

```
a.banner {
    position: relative;
    background:
        url(../images/banner-arrow.png) no-repeat -100px 140px,
        url(../images/banner-photo.png) no-repeat -40px 220px,
        url(../images/banner-09.png) no-repeat -20px -380px,
        url(../images/banner-bg.png) no-repeat 0 0;
-webkit-transition: all .2s ease-in .2s;
    -moz-transition: all .2s ease-in .2s;
    -o-transition: all .2s ease-in .2s;
    -ms-transition: all .2s ease-in .2s;
    transition: all .2s ease-in .2s;
}
.modern a.banner:hover, a.banner:focus {
    background-position:
        20px 140px,
        -40px 20px,
        -20px -90px,
        0 0;
}
.modern .banner-logo {
    position: absolute;
    top: 20px;
    left: 270px;
    -webkit-transition: all .4s ease-out .3s;
    -moz-transition: all .4s ease-out .3s;
    -o-transition: all .4s ease-out .3s;
    -ms-transition: all .4s ease-out .3s;
    transition: all .4s ease-out .3s;
}
.modern a.banner:hover .banner-logo,
.modern a.banner:focus .banner-logo {
        left: 540px;
}
.modern .banner-desc {
    opacity: 0;
    position: absolute;
    top: 39px;
    left: 170px;
    font: 39px/1.1 "NanumPenWeb", "方正静蕾简体", "Nanum Pen Script";
    color: #4ec1cd;
    -webkit-transition: all .4s ease-out .3s;
    -moz-transition: all .4s ease-out .3s;
    -o-transition: all .4s ease-out .3s;
    -ms-transition: all .4s ease-out .3s;
    transition: all .4s ease-out .3s;
}
```

最后使用JQuery播放声音文件

```
/* banner.js - Banner设计脚本, 2012 © yamoo9.com
---------------------------------------------------------------- */
```

```
;(function($){
    $(function() { // $(document).ready(); 文档准备好后运行

        var banner_audio= new Audio(),         // 创建Audio.
             webm = isSupportWebM();    // 检查是否支持webm格式
        banner_audio.src = 'media/banner_sound.mp3';
        /*if(webm) {    //支持webm格式
            banner_audio.src = 'media/banner_sound.webm';
        } else {    // 不支持webm格式
            banner_audio.src = 'media/banner_sound.mp3';
        };*/
        $('.banner')
            .bind('mouseover focusin', function() { // 当发生MouseOver，FocusIn事件时
调用处理函数
                banner_audio.load(); // 加载audio
                banner_audio.play(); // 播放audio
            })
            .bind('mouseout focusout', function() { // 当发生MouseOut，FocusOut事件时
调用处理函数
                banner_audio.pause();              // 暂停audio
                banner_audio.currentTime = 0;     // 初始化audio播放位置
            });

    });
})(window.jQuery);
// 检测是否webm格式的函数
function isSupportWebM() {
    var tester = document.createElement('audio');
    return !!tester.canPlayType('audio/webm');
};
```

最后的完成作品：
http://pan.baidu.com/s/1hsCWACs

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5456859>

HTML5气泡悬浮框(已经加上完整文件)

2016年5月6日　　13:30

源文件链接：
http://pan.baidu.com/s/1pKHlNSn

设计气泡悬浮框

1.在网页设计中，气泡悬浮框常常用于页面中为某些对象显示提示信息，恰当地使用气泡悬浮框能够使网页布局更加完美，使网页看上去更漂亮、美观；

2.一般而言，替换文本使用alt属性来呈现，说明性文本通过title属性来实现，这两个属性是HTML默认提供的功能，在网页设计中使用它们，容易引起用户的反感；

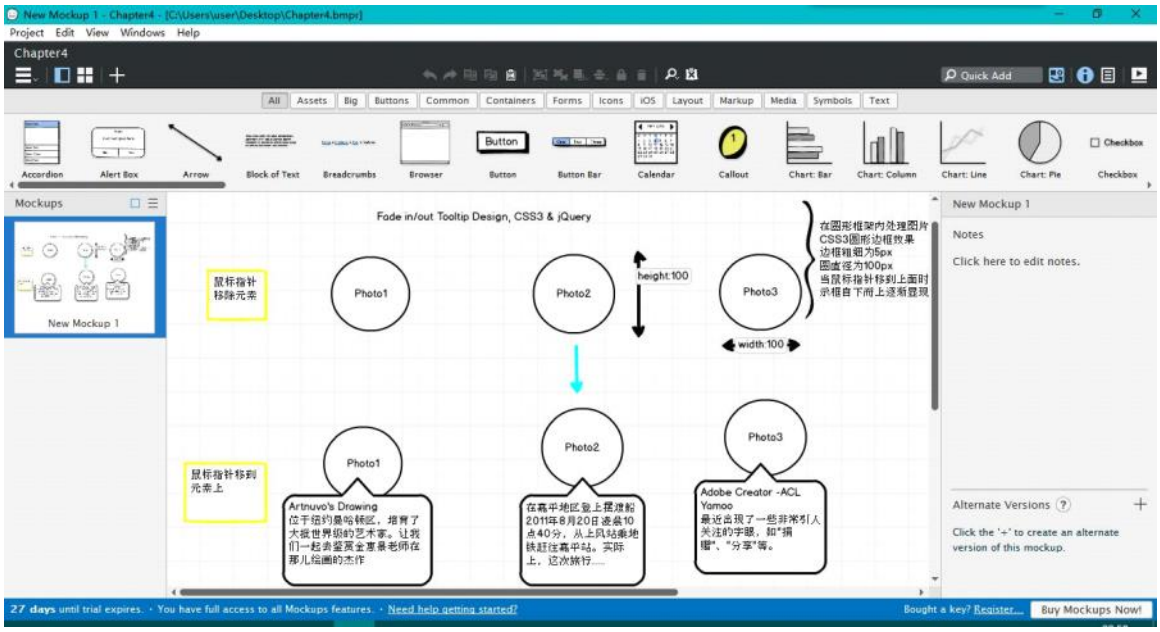
3.在这种情况下，使用气泡悬浮框能够产生不一样的视觉体验。

下面使用气泡悬浮框来设计网页中的替代文本与说明文本。

设计过程：

（一）设计网页布局

使用Balsamiq Mockups工具将网页布局描绘出来，如下图所示。



(二)编写HTML5代码

我们将标题设置为: Fade in/out Tooltip Design, CSS3 & jQuery

但是在这里需要注意"字符实体"

如果要显示& 那么实体名称为& 实体编号&

其他字符实体见下表：

显示结果	描述	实体名称	实体编号
	空格		
<	小于号	&lt;	&#60;
>	大于号	&gt;	&#62;
&	和号	&amp;	&#38;
"	引号	&quot;	&#34;
'	撇号	' (IE不支持)	&#39;
¢	分	&cent;	&#162;
£	镑	&pound;	&#163;
¥	日圆	&yen;	&#165;
€	欧元	&euro;	&#8364;
§	小节	&sect;	&#167;
©	版权	&copy;	&#169;
®	注册商标	&reg;	&#174;
™	商标	&trade;	&#8482;
×	乘号	&times;	&#215;
÷	除号	&divide;	&#247;

下面添加<a>并设置class 和 href.在<a>标签内插入 标签

```
<a class="tooltip photo1" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">在嘉平地区登上摆渡船</h4>
        <p class="tooltip-desc">2011年8月20日清晨10点40分，从上风站乘地铁赶往嘉平站。事实上，这次旅行.....</p>
    </div>
</a>
```

添加其他的<a>标签：

```
<a class="tooltip photo1" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">在嘉平地区登上摆渡船</h4>
        <p class="tooltip-desc">2011年8月20日清晨10点40分，从上风站乘地铁赶往嘉平站。事实上，这次旅行.....</p>
    </div>
</a>
</li>
<li>
<a class="tooltip photo2" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">Artnuvo's drawing</h4>
        <p class="tooltip-desc">Art Student Lengue of NewYork位于纽约曼哈顿区，培养了大批世界级的艺术家。让我们一起
            鉴赏金惠景老师在那儿绘画的杰作!</p>
    </div>
</a>
</li>
<li>
<a class="tooltip photo3" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">Adobe Creator -ACL Yamoo</h4>
        <p class="tooltip-desc">最近出现了一些非常引人关注的字眼，如"赠"、"分享"等。ACL Yamoo积极响应这场运动，
            开放了“开放知识讲座”项目，跟大家一起分享知识</p>
```

```
    </div>
</a>
```

（三）编写CSS3样式表

1.控制body样式

```
body{
    padding: 150px;
    background: #2b2b2b url(../images/bg_tile.jpg);
}
```

2.设置标题字体样式

```
h1{
    margin-bottom: 40px;
    font-family: 'Lato',Sans-Serief;
    color:#fff;
}
```

同时需要在CSS代码前加上字体应用

```
@import url(http://fonts.googleapis.com/css?family=Lato:100);
```

3.控制图片样式

```
a.tooltip{
    position: relative;
    display:block;
    width:100px;
    height:100px;
    border:5px solid #4b4b4b;
    background:#fff no-repeat center;
}
```

同时设置.gallery-nav内的li元素

```
.gallery-nav li{
    float:left;
    margin-right:100px;
}
```

下面分别为控制a.tooltip、a.tooltip.photo1(photo2 / photo3)的样式

```
a.tooltip{
    border-radius:55px;
    -webkit-border-radius:55px;
    -khtml-border-radius:55px;
    -moz-border-radius:55px;
}
a.tooltip.photo1{
    background-image: url(../images/sussjini-bbo.jpg);
}
a.tooltip.photo2{
    background-image: url(../images/khk-artwork.png);
}
```

```
a.tooltip.photo3{
    background-image: url(../images/interview-yamoo9.png);
}
```

4.控制气泡悬浮框1：基本样式、位置、添加圆角与尾巴并制作Transition动画

```
a.tooltip .tooltip-box {
    opacity:0;
    position: absolute;
    left:50%;
    bottom:100px;
    width:20em;
    margin-left:-10.4em;
    padding:.8em;
    background:#111;
    -webkit-border-radius:15px 0px;
    -khtml-border-radius:15px 0px;
    -o-border-radius:15px 0px;
    border-radius:15px 0px;

    -webkit-transition:all .4s ease-in .3s;
    -moz-transition:all .4s ease-in .3s;
    -o-transition:all .4s ease-in .3s;
    -ms-transition:all .4s ease-in .3s;
    transition:all .4s ease-in .3s;
}
a.tooltip:hover .tooltip-box,
a.tooltip:focus .tooltip-box {
    opacity: 1;
    bottom: 90px;
}
a.tooltip .tooltip-box:before {
    content: '';
    position: absolute;
    bottom: -10px;
    left: 120px;
    border-top: 10px solid #111;
    border-left: 10px solid transparent;
    border-right: 10px solid transparent;
}
a.tooltip .tooltip-title {
    color:#fff;
}
a.tooltip.tooltip-desc{
    margin-bottom:0;
    font-size:11px;
    text-align:justify;
    color:#bcbcbc;
}
```

下面给出完整代码：

```
<!DOCTYPE html>
<!--[if IE 6]><html lang="zh" class="no-js old ie6"><![endif]-->
<!--[if IE 7]><html lang="zh" class="no-js old ie7"><![endif]-->
<!--[if IE 8]><html lang="zh" class="no-js old ie8"><![endif]-->
<!--[if IE 9]><html lang="zh" class="no-js modern ie9"><![endif]-->
<!--[if !IE]><!--><html lang="zh" class="no-js modern"><!--<![endif]-->
<head>
<meta charset="utf-8" />
<title>CSS3 Tooltip Design - 淡入/淡出提示工具设计</title>
<link rel="stylesheet" href="css/tooltip.css" />
<script src="js/jquery.min.js"></script>
<script src="js/tooltip.js"></script>
</head>
```

```
<body>
<h1> Fade in/out Tooltip Design, CSS3 &amp;jQuery</h1>
<ul class="gallery-nav">
<li>
<a class="tooltip photo1" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">在嘉平地区登上摆渡船</h4>
        <p class="tooltip-desc">2011年8月20日清晨10点40分，从上风站乘地铁赶往嘉平站。事实上，这次旅行.....</p>
    </div>
</a>
</li>
<li>
<a class="tooltip photo2" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">Artnuvo's drawing</h4>
        <p class="tooltip-desc">Art Student Lengue of NewYork位于纽约曼哈顿区，培养了大批世界级的艺术家。让我们一起
            鉴赏金惠景老师在那儿绘画的杰作!</p>
    </div>
</a>
</li>
<li>
<a class="tooltip photo3" href="http://yamoo9.com/?p=699">
    <div class="tooltip-box">
        <h4 class="tooltip-title">Adobe Creator -ACL Yamoo</h4>
        <p class="tooltip-desc">最近出现了一些非常引人关注的字眼，如"赠"、"分享"等。ACL Yamoo积极响应这场运动，
            开放了“开放知识讲座”项目，跟大家一起分享知识</p>
    </div>
</a>
</li>
</body>
</html>
```

完整CSS代码：

```
@charset "utf-8";
@import "reset.css";
@import url(http://fonts.googleapis.com/css?family=Lato:100);
/* tooltip.css - ToolTip设计样式, 2012 © yamoo9.com
---------------------------------------------------------------------- */
body{
    padding: 150px;
    background: #2b2b2b url(../images/bg_tile.jpg);
}
h1{
    margin-bottom: 40px;
    font-family: 'Lato',Sans-Serief;
    color:#fff;
}
.gallery-nav li{
    float:left;
    margin-right:100px;
}
a.tooltip{
    position: relative;
    display:block;
    width:100px;
    height:100px;
    border:5px solid #4b4b4b;
    background:#fff no-repeat center;
    background-size:cover;

    border-radius:55px;
    -webkit-border-radius:55px;
    -khtml-border-radius:55px;
    -moz-border-radius:55px;

    -webkit-transition:all .4s ease-in .3s;
```

```
    -moz-transition:all .4s ease-in .3s;
    -o-transition:all .4s ease-in .3s;
    -ms-transition:all .4s ease-in .3s;
    transition:all .4s ease-in .3s;
}
a.tooltip:hover,
a.tooltip:focus{
    border-color:#fff;
}
a.tooltip .tooltip-box {
    opacity:0;
    position: absolute;
    left:50%;
    bottom:100px;
    width:20em;
    margin-left:-10.4em;
    padding:.8em;
    background:#111;
    -webkit-border-radius:15px 0px;
    -khtml-border-radius:15px 0px;
    -o-border-radius:15px 0px;
    border-radius:15px 0px;

    -webkit-transition:all .4s ease-in .3s;
    -moz-transition:all .4s ease-in .3s;
    -o-transition:all .4s ease-in .3s;
    -ms-transition:all .4s ease-in .3s;
    transition:all .4s ease-in .3s;
}
a.tooltip:hover .tooltip-box,
a.tooltip:focus .tooltip-box {
    opacity: 1;
    bottom: 90px;
}
a.tooltip .tooltip-box:before {
    content: '';
    position: absolute;
    bottom: -10px;
    left: 120px;
    border-top: 10px solid #111;
    border-left: 10px solid transparent;
    border-right: 10px solid transparent;
}
a.tooltip .tooltip-title {
    color:#fff;
}
a.tooltip.tooltip-desc{
    margin-bottom:0;
    font-size:11px;
    text-align:justify;
    color:#bcbcbc;
}
a.tooltip.photo1{
    background-image: url(../images/sussjini-bbo.jpg);
}
a.tooltip.photo2{
    background-image: url(../images/khk-artwork.png);
}
a.tooltip.photo3{
    background-image: url(../images/interview-yamoo9.png);
}
.clearfix:after{
    content:"";
    display: block;
    clear:both;
}
.ie6.clearfix{height:1px;}
.ie7.clearfix{min-height:1px;}
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5460180>

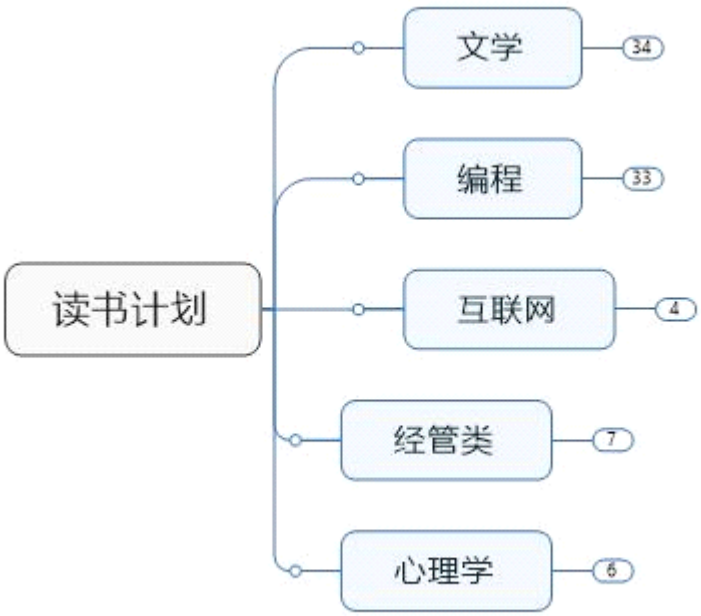
未来读书计划

2016年5月6日　　13:30

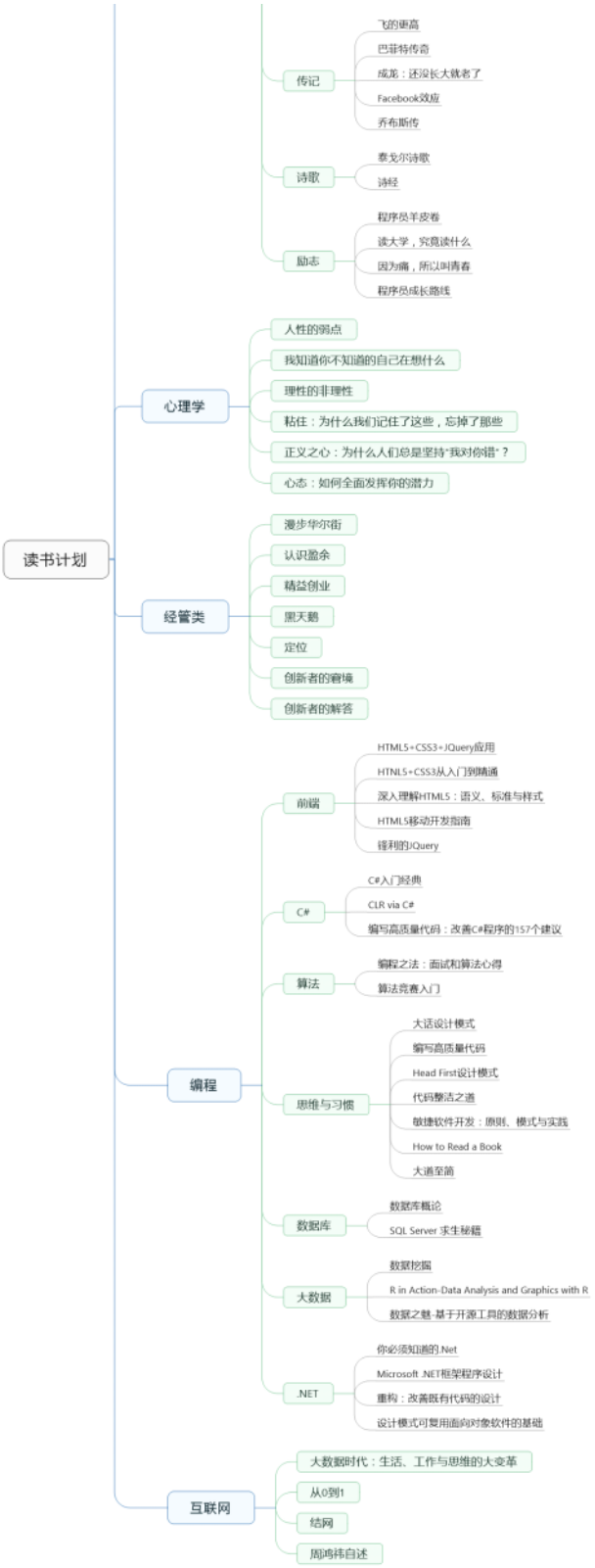
也许是这学期才开始进入真正的读书时间，这学期我才真正把读书当做一种爱好，一种乐此不彼的习惯。在这里十分感谢东野圭吾的小说，也许这些小说是我进入阅读的一把钥匙。读过的书越多，发觉自己知道的越少，从而更加渴望读到更多的好书。上课的时候老师曾经举过一个例子，一个人的知识以及对世界的认识就好比一个圆，圆越大那么和周围的接触越大，圆越小接触就越少，那些认为什么都懂得反而是小圆，那些认为什么什么都不懂的才是大圆。也许这个例子不是那么准确，但在一定的程度上向我们表述了掌握的越多，困惑也就越多。因此个人认为**“活到老，学到老”**可以作为人生的信条。

读书盘点

下面是用MindManager整理的读书条目，包含读过的和正在读的，一共83本，文学类和编程几乎各占40%。



下面是具体的书目：



再一次地感谢小说家**东野圭吾（ひがしの けいご，Higashino Keigo）**,我不是否认其他作家关于悬疑推理的作品，只是因为让我沉醉其中的第一个作品是他的,我担心看了其他的作品之后会和我已经形成的认识产生矛盾,也许正是这样的心态,对于他的作品才能更加坚定,同样也是因为对于他的作品的热爱感染了自身的阅读细胞。这也就是为何这学期是真正的读书时间。只怪自己不会日语,如果可能希望能够读懂日文版的作品。

身在理工科的程序猿除了提高专业知识之外，个人的文化修养还是要注重起来。可以看到编程以及文学所占比例的确很大。

现在的我对于阅读满满的都是期待。O(∩_∩)O~~

现在的我对于阅读满满的都是期待。O(∩_∩)O~~

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5463846>

FFT教你做乘法（FFT傅里叶变换）

2016年5月6日　　13:31

题目来源：https://biancheng.love/contest/41/problem/C/index

FFT教你做乘法

题目描述

给定两个8进制正整数A和B（A和B均小于10000位），请利用离散傅里叶变换计算A与B的乘积。

输入

多组测试数据（组数不超过100）每组测试数据只有一行，包含两个正整数A和B。

输出

对于每组数据，输出一行，为A和B的乘积。

输入样例

```
1 7
2 17
```

输出样例

```
7
36
```

解题思路：

推荐博客（有助于理解FFT）：http://blog.jobbole.com/58246/

推荐博客（FFT之大数相乘）：http://www.cnblogs.com/lsx54321/archive/2012/07/20/2601632.html

给出代码：

```cpp
#include <bits/stdc++.h>
#define N 505050

using namespace std;

const double PI=acos(-1.0);
struct Vir
{
    double re,im;
    Vir(double _re=0.,double _im=0.):re(_re),im(_im) {}
    Vir operator*(Vir r)
    {
        return Vir(re*r.re-im*r.im,re*r.im+im*r.re);
    }
    Vir operator+(Vir r)
    {
        return Vir(re+r.re,im+r.im);
    }
    Vir operator-(Vir r)
    {
        return Vir(re-r.re,im-r.im);
    }
};


void bit_rev(Vir *a,int loglen,int len)
{
    for(int i=0; i<len; ++i)
    {
        int t=i,p=0;
        for(int j=0; j<loglen; ++j)
        {
            p<<=1;
            p=p|(t&1);
            t>>=1;
        }
        if(p<i)
        {
            Vir temp=a[p];
            a[p]=a[i];
            a[i]=temp;
        }
    }
}
void FFT(Vir *a,int loglen,int len,int on)
{
    bit_rev(a,loglen,len);

    for(int s=1,m=2; s<=loglen; ++s,m<<=1)
    {
        Vir wn=Vir(cos(2*PI*on/m),sin(2*PI*on/m));
        for(int i=0; i<len; i+=m)
        {
            Vir w=Vir(1.0,0);
            for(int j=0; j<m/2; ++j)
            {
                Vir u=a[i+j];
                Vir v=w*a[i+j+m/2];
                a[i+j]=u+v;
                a[i+j+m/2]=u-v;
                w=w*wn;
            }
        }
    }
    if(on==-1)
    {
        for(int i=0; i<len; ++i) a[i].re/=len,a[i].im/=len;
    }
}
char a[N*2],b[N*2];
Vir pa[N*2],pb[N*2];
int ans[N*2];

int main ()
{
```

```
76      while(scanf("%s%s",a,b)!=EOF)
77      {
78          int lena=strlen(a);
79          int lenb=strlen(b);
80          int n=1,loglen=0;
81          while(n<lena+lenb)
82          {
83              n<<=1,loglen++;
84          }
85          for(int i=0,j=lena-1; i<n; ++i,--j)
86              pa[i]=Vir(j>=0?a[j]-'0':0.,0.);
87          for(int i=0,j=lenb-1; i<n; ++i,--j)
88              pb[i]=Vir(j>=0?b[j]-'0':0.,0.);
89          for(int i=0; i<=n; ++i)
90          {
91              ans[i]=0;
92          }
93          FFT(pa,loglen,n,1);
94          FFT(pb,loglen,n,1);
95          for(int i=0; i<n; ++i)
96              pa[i]=pa[i]*pb[i];
97          FFT(pa,loglen,n,-1);
98
99          for(int i=0; i<n; ++i) ans[i]=pa[i].re+0.5;
100         for(int i=0; i<n; ++i) ans[i+1]+=ans[i]/8,ans[i]%=8;
101
102         int pos=lena+lenb-1;
103         for(; pos>0&&ans[pos]<=0; --pos) ;
104         for(; pos>=0; --pos) printf("%d",ans[pos]);
105         printf("\n");
106     }
107     return 0;
108 }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5074750>

寻找最远点对（凸包求解）

2016年5月6日　　13:31

题目来源：https://biancheng.love/contest/41/problem/B/index

寻找最远点对

题目描述

TD走廊里有一关“勇闯梅花桩”，水面上稀稀落落地立着几根柱子。Nova君自认为轻功不错，觉得可以在任意两根柱子之间跳跃，现在他想挑战一次跨越距离最远的两根柱子。请问，最远距离是多少？（由于木桩以横纵坐标形式给出，为了计算方便，避免求平方根，答案只需给出距离的平方即可）

输入

多组测试数据（组数不超过10），对于每组数据，第一行为一个正整数N，代表梅花桩的个数，接下来N行，每行两个正整数xi， yi分别代表第 i 根桩子的横纵坐标。（数据在INT范围内）

输出

对于每组数据，输出一行，为距离最远的两根柱子的距离的平方。

输入样例

```
3
1 1
1 2
0 0
```

输出样例

```
5
```

解题思路：

求出最远的点对，可以化为求出对应的凸包上最远的两个点。

平面凸包 ：

定义： 对一个简单多边形来说，如果给定其边界上或内部的任意两个点，连接这两个点的线段上的所有点都被包含在该多边形的边界上或内部的话，则该多边形为凸多边形 。

在解决平面凸包下面介绍了两种算法：

一、 Graham扫描法，运行时间为O(nlgn)。

二、 Jarvis步进法，运行时间为O(nh),h为凸包中的顶点数。

推荐博客：
http://blog.csdn.net/bone_ace/article/details/46239187
推荐博客：
http://www.cnblogs.com/jbelial/archive/2011/08/05/2128625.html
给出代码：

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#define INF -110

using namespace std;
long long i,j,k,n,top,ans;

struct Node
{
 int x;
 int y;
}no[1000010],stack[1000010];

long long dis(Node p1,Node p2)
{
    return (p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y);
}

long long mult(Node p1,Node p2,Node p0)
{
    return ((p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y));
}

long long cmp(Node a,Node b)
{
    if(mult(a,b,no[0])>0)
        return 1;
    else if(mult(a,b,no[0])==0&&dis(a,no[0])<dis(b,no[0]))
        return 1;
    return 0;
}

void work()
{
    k=0;
    for(i=1; i<n; i++)
    {
        if(no[k].y>no[i].y || ((no[k].y == no[i].y) && no[k].x > no[i].x))
            k = i;
    }
    Node tmp;
    tmp = no[0];
    no[0] = no[k];
    no[k] = tmp;
    sort(no+1,no+n,cmp);
    top = 2;
```

```
48      stack[0] = no[0];
49      stack[1] = no[1];
50      stack[2] = no[2];
51      for(i=3; i<n; i++)
52      {
53          while(top>1 && mult(no[i],stack[top],stack[top-1])>=0)
54              top--;
55          stack[++top] = no[i];
56      }
57  }
58
59  int main()
60  {
61      while(~scanf("%lld",&n))
62      {
63          for(i=0; i<n; i++)
64              scanf("%lld%lld",&no[i].x,&no[i].y);
65          work();
66          ans=INF;
67          for(i=0; i<=top; i++)
68          {
69              for(j=i+1; j<=top; j++)
70              {
71                  if(ans<dis(stack[i],stack[j]))
72                      ans=dis(stack[i],stack[j]);
73              }
74          }
75          printf("%lld\n",ans);
76      }
77      return 0;
78  }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5074741>

捡火柴的Nova君（n个线段相交问题）

2016年5月6日　　13:32

题目来源：https://biancheng.love/contest-ng/index.html#/41/problems

捡火柴的Nova君
题目描述

南方没暖气，怕冷的的宝宝们只能用火柴取暖。然而Nova君害怕火烧到手指头，当木头梗还有一大截的时候就慌忙把火柴丢到地上踩灭了，没多久，地上就七零八落，躺着一堆木棍，有得木梗还盖在了别的上面。现在Nova君打算收拾残局，他准备一根一根捡起火柴，但他很怕多花力气，所以决定优先捡起没有被盖住的火柴梗（因为被盖住的捡起来更费力嘛），所以现在Nova君，有多少根火柴是没有被盖住的呢？

输入

多组测试数据（组数不超过100），对于每组数据，第一行为一个正整数N，表示火柴梗的根数，接下来N行，每行四个浮点数a1,a2,b1,b2，分别表示火柴梗两个端点的横纵坐标。（请用double数据类型，保证输入数据合法）

输出

对于每组数据，输出一行，表示没有被盖住的火柴序号（输出的相对顺序与输入时保持一致）

输入样例

```
2
1 1 2 2
1 1 3 3
3
0 0 1 1
1 0 2 1
2 0 3 1
```

输出样例

```
2
1 2 3
```

Hint

抽象出的线段，只要有点重合，就算是覆盖了

解题思路：

一根火柴两个端点，两个端点组成一个线段。意思就是找到没有被覆盖的火柴序号。

注意：火柴序号即火柴输入的顺序（1,2,3。。。）

姿势很多，给出结构体代码：

```
#include <bits/stdc++.h>
#define eps 1e-6
#define MAX 100010

using namespace std;

int sgn(double x)
{
    if(fabs(x)<eps)
        return 0;
    if(x<0)
        return -1;
    else return 1;
}

struct Point
{
```

```
18      double x,y;
19      Point() {}
20      Point(double x1,double y1)
21      {
22          x=x1;
23          y=y1;
24      }
25      Point operator -(Point b)
26      {
27          return Point(x-b.x,y-b.y);
28      }
29      double operator ^(const Point b)
30      {
31          return x*b.y-y*b.x;
32      }
33      double operator *(Point b)
34      {
35          return x*b.x+y*b.y;
36      }
37  };
38
39  struct Line
40  {
41      Point start,over;
42      Line() {}
43      Line(Point start1,Point over1)
44      {
45          start=start1;
46          over=over1;
47      }
48  };
49
50  bool cs(Line l1,Line l2)
51  {
52      return
53          ( max(l1.start.x,l1.over.x)>=min(l2.start.x,l2.over.x)
54          &&max(l2.start.x,l2.over.x)>=min(l1.start.x,l1.over.x)
55          &&max(l1.start.y,l1.over.y)>=min(l2.start.y,l2.over.y)
56          &&max(l2.start.y,l2.over.y)>=min(l1.start.y,l1.over.y)
57          &&sgn((l2.start-l1.start)^(l1.over-l1.start))*sgn((l2.over-
l1.start)^(l1.over-l1.start))<=0
58          &&sgn((l1.start-l2.start)^(l2.over-l2.start))*sgn((l1.over-
l2.start)^(l2.over-l2.start))<=0 );
59  }
60
61  Line line[MAX];
62  bool flag[MAX];
63  int main()
64  {
65      int n,i,j;
66      double x1,y1,x2,y2;
67      while(~scanf("%d",&n))
68      {
69          for(i=1;i<=n;i++)
70          {
71              scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);
72              line[i]=Line(Point(x1,y1),Point(x2,y2));
73              flag[i]=true;
74          }
75          for(i=1;i<=n;i++)
76          {
77              for(j=i+1;j<=n;j++)
78                  if(cs(line[i],line[j]))
79                  {
80                      flag[i]=false;
81                      break;
82                  }
83          }
84          for(i=1;i<=n;i++)
85          {
86              if(flag[i])
87              {
88                  printf("%d ",i);
89              }
90          }
```

```
91          printf("\n");
92      }
93      return 0;
94  }
```

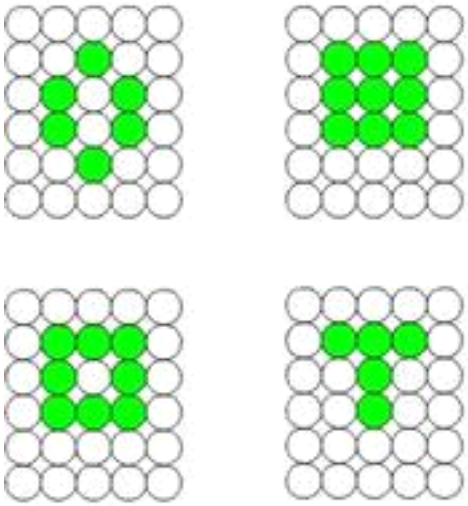
来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5074730>

生命游戏/Game of Life的Java实现（转）

2016年5月6日　　13:32

首先简单介绍一下《生命游戏》

生命游戏其实是一个零玩家游戏。它包括一个二维矩形世界，这个世界中的每个方格居住着一个活着的或死了的细胞。一个细胞在下一个时刻生死取决于相邻八个方格中活着的或死了的细胞的数量。如果相邻方格活着的细胞数量过多，这个细胞会因为资源匮乏而在下一个时刻死去；相反，如果周围活细胞过少，这个细胞会因太孤单而死去。具体如下图：



每个格子的生死遵循下面的原则：

1. 如果一个细胞周围有3个细胞为生（一个细胞周围共有8个细胞），则该细胞为生（即该细胞若原先为死，则转为生，若原先为生，则保持不变）。

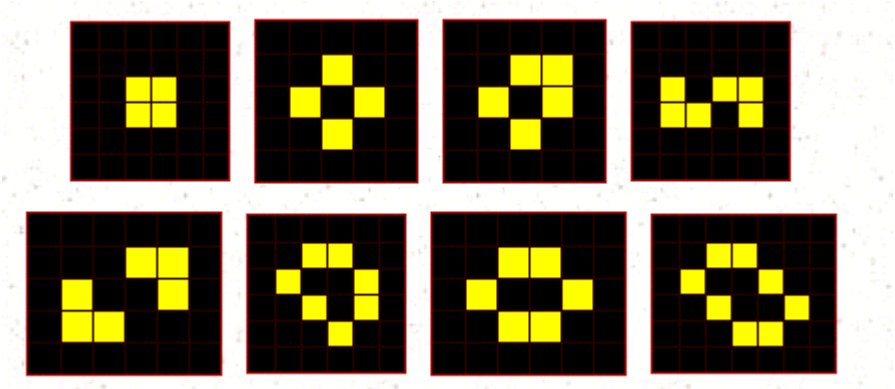
2. 如果一个细胞周围有2个细胞为生，则该细胞的生死状态保持不变；

3. 在其它情况下，该细胞为死（即该细胞若原先为生，则转为死，若原先为死，则保持不变设定图像中每个像素的初始状态后依据上述的游戏规则演绎生命的变化，由于初始状态和迭代次数不同，将会得到令人叹服的优美图案）。

图案：

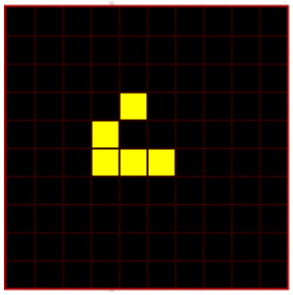
有三类图案：

第一类这种图案是固定不变的：



第二类：是按规律变化的：

第三类：是不断变化和移动的，但是有一定的周期性，最著名的就是 滑翔机：



Java实现代码：

```
1 package com.cisco.gendwang;
2
3
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6
7
8 import javax.swing.JFrame;
9 import javax.swing.JMenu;
10 import javax.swing.JMenuBar;
11 import javax.swing.JMenuItem;
12
13
14 public class LifeGame extends JFrame
15 {
16     private final World world;
17
18     public LifeGame(int rows, int columns)
19     {
20         world = new World(rows, columns);
21         new Thread(world).start();
22         add(world);
23     }
24
25     public static void main(String[] args)
26     {
27         LifeGame frame = new LifeGame(40, 50);
28
29         JMenuBar menu = new JMenuBar();
30         frame.setJMenuBar(menu);
31
32         JMenu options = new JMenu("Options");
33         menu.add(options);
34
35         JMenuItem arrow = options.add("Arrow");
36         arrow.addActionListener(frame.new ArrowActionListener());
37
38         JMenuItem square = options.add("Square");
39         square.addActionListener(frame.new SquareActionListener());
40
41         JMenu help = new JMenu("Help");
42         menu.add(help);
43
44         frame.setLocationRelativeTo(null);
45         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46         frame.setSize(1007, 859);
47         frame.setTitle("Game of Life");
48         frame.setVisible(true);
49         frame.setResizable(false);
50     }
51
52     class ArrowActionListener implements ActionListener
53     {
54         public void actionPerformed(ActionEvent e)
55         {
56             world.setArrow();
57         }
58     }
59
60     class SquareActionListener implements ActionListener
61     {
```

```
62          public void actionPerformed(ActionEvent e)
63          {
64              world.setSquare();
65          }
66      }
67  }
```

View Code

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5071684>

如果看了此文你还不懂傅里叶变换，那就过来掐死我吧【完整版】(转)

2016年5月6日　　13:32

转载地址：http://blog.jobbole.com/70549/

要让读者在不看任何数学公式的情况下理解傅里叶分析。

傅里叶分析不仅仅是一个数学工具，更是一种可以彻底颠覆一个人以前世界观的思维模式。但不幸的是，傅里叶分析的公式看起来太复杂了，所以很多大一新生上来就懵圈并从此对它深恶痛绝。老实说，这么有意思的东西居然成了大学里的杀手课程，不得不归咎于编教材的人实在是太严肃了。（您把教材写得好玩一点会死吗？会死吗？）所以我一直想写一个有意思的文章来解释傅里叶分析，有可能的话高中生都能看懂的那种。所以，不管读到这里的您从事何种工作，我保证您都能看懂，并且一定将体会到通过傅里叶分析看到世界另一个样子时的快感。至于对于已经有一定基础的朋友，也希望不要看到会的地方就急忙往后翻，仔细读一定会有新的发现。

————以上是定场诗————

下面进入正题：

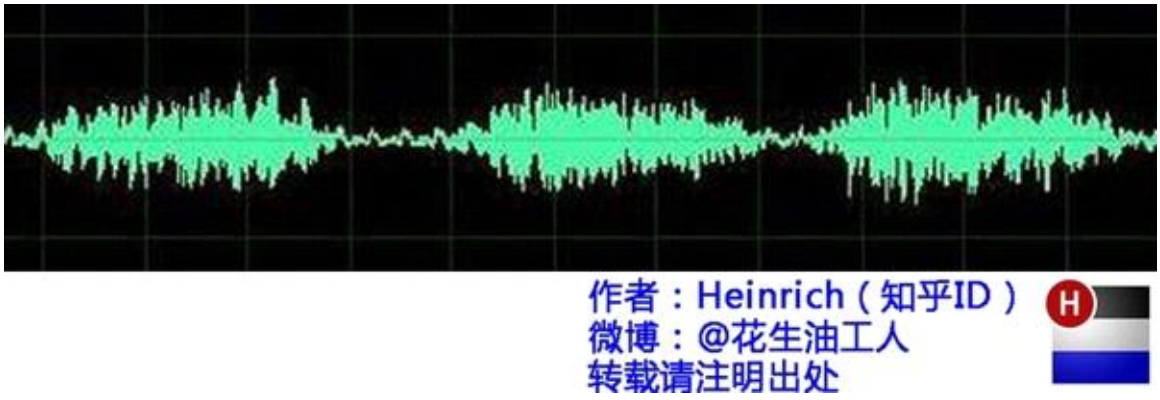
抱歉，还是要啰嗦一句：其实学习本来就不是易事，我写这篇文章的初衷也是希望大家学习起来更加轻松，充满乐趣。但是千万！千万不要把这篇文章收藏起来，或是存下地址，心里想着：以后有时间再看。这样的例子太多了，也许几年后你都没有再打开这个页面。无论如何，耐下心，读下去。这篇文章要比读课本要轻松、开心得多……

一、什么是频域

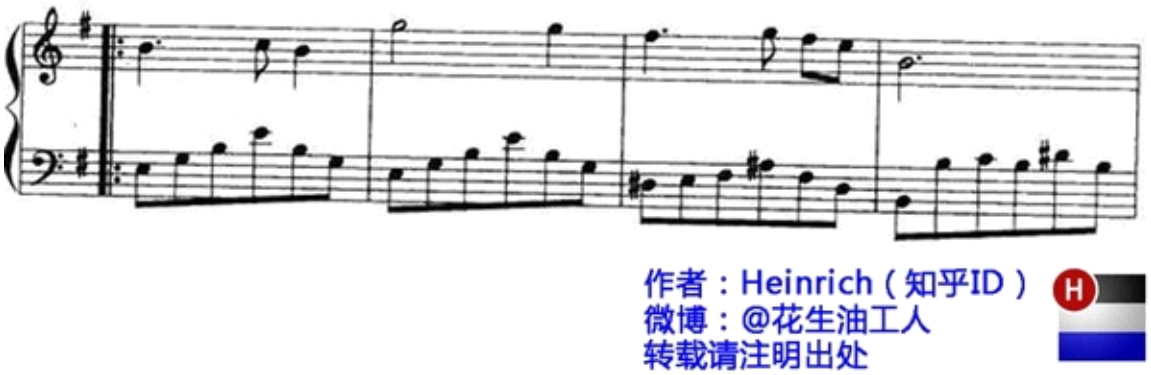
从我们出生，我们看到的世界都以时间贯穿，股票的走势、人的身高、汽车的轨迹都会随着时间发生改变。这种以时间作为参照来观察动态世界的方法我们称其为时域分析。而我们也想当然的认为，世间万物都在随着时间不停的改变，并且永远不会静止下来。但如果我告诉你，用另一种方法来观察世界的话，你会发现**世界是永恒不变的**，你会不会觉得我疯了？我没有疯，这个静止的世界就叫做频域。

先举一个公式上并非很恰当，但意义上再贴切不过的例子：

在你的理解中，一段音乐是什么呢？



这是我们对音乐最普遍的理解，一个随着时间变化的震动。但我相信对于乐器小能手们来说，音乐更直观的理解是这样的：



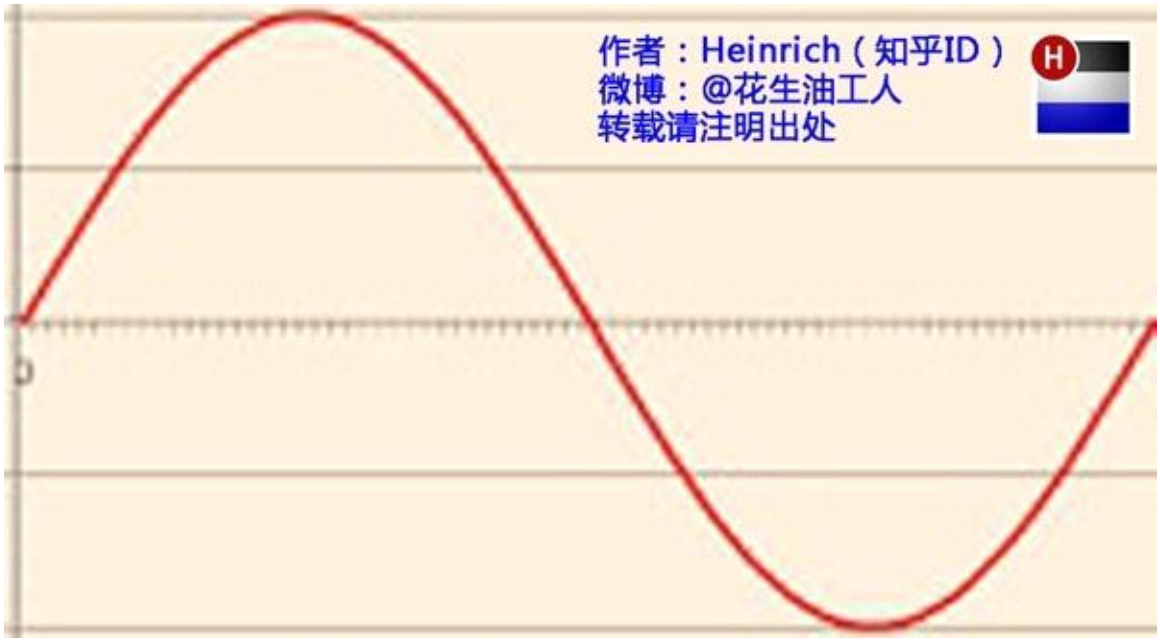
好的！下课，同学们再见。

是的，其实这一段写到这里已经可以结束了。上图是音乐在时域的样子，而下图则是音乐在频域的样子。所以频域这一概念对大家都从不陌生，只是从来没意识到而已。

现在我们可以回过头来重新看看一开始那句痴人说梦般的话：世界是永恒的。

将以上两图简化：

时域：



频域：

在时域，我们观察到钢琴的琴弦一会上一会下的摆动，就如同一支股票的走势；而在频域，只有那一个永恒的音符。

所以

你眼中看似落叶纷飞变化无常的世界，实际只是躺在上帝怀中一份早已谱好的乐章。

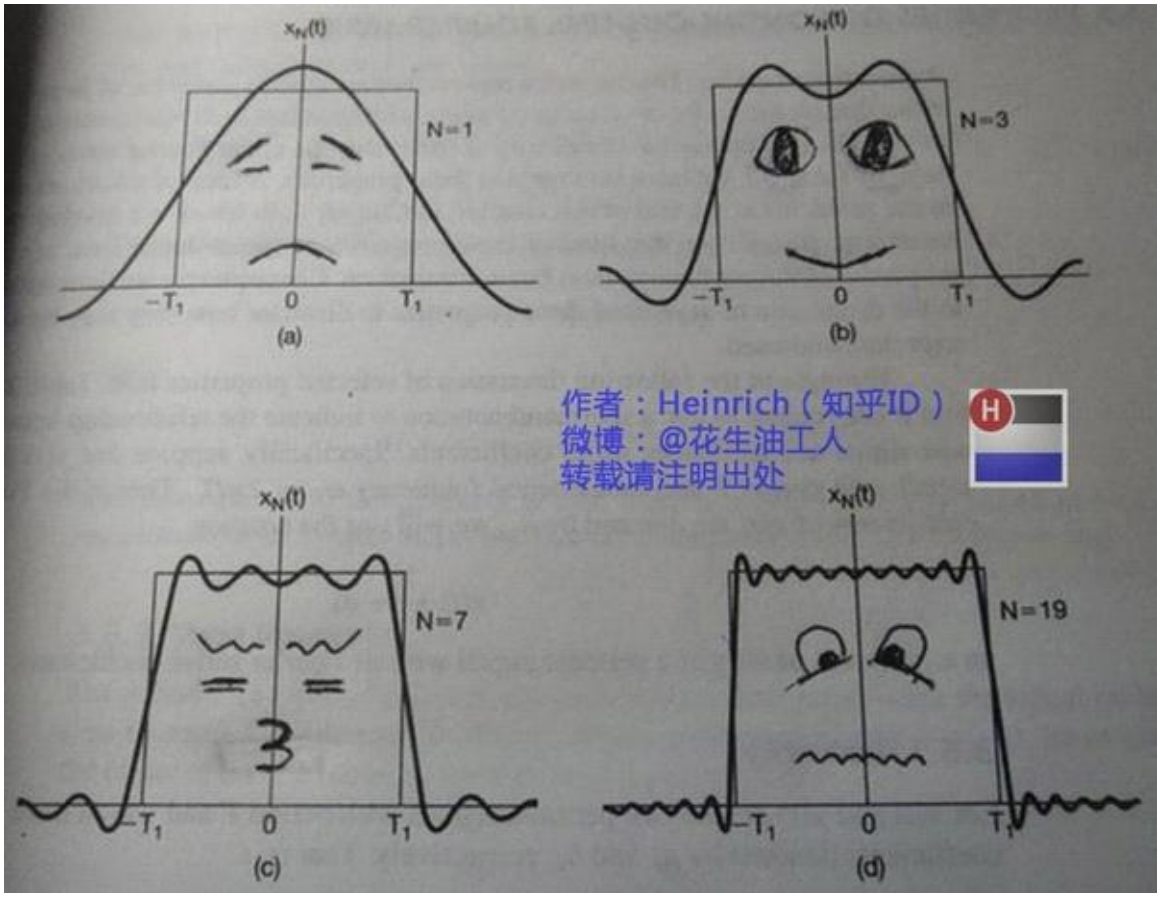
抱歉，这不是一句鸡汤文，而是黑板上确凿的公式：傅里叶同学告诉我们，任何周期函数，都可以看作是不同振幅，不同相位正弦波的叠加。在第一个例子里我们可以理解为，利用对不同琴键不同力度，不同时间点的敲击，可以组合出任何一首乐曲。

而贯穿时域与频域的方法之一，就是传中说的傅里叶分析。傅里叶分析可分为傅里叶级数（Fourier Serie）和傅里叶变换(Fourier Transformation)，我们从简单的开始谈起。

二、傅里叶级数(Fourier Series)的频谱

还是举个栗子并且有图有真相才好理解。

如果我说我能用前面说的正弦曲线波叠加出一个带 90 度角的矩形波来，你会相信吗？你不会，就像当年的我一样。但是看看下图：



第一幅图是一个郁闷的正弦波 cos（x）

第二幅图是 2 个卖萌的正弦波的叠加 cos (x) +a.cos (3x)

第三幅图是 4 个发春的正弦波的叠加

第四幅图是 10 个便秘的正弦波的叠加

随着正弦波数量逐渐的增长，他们最终会叠加成一个标准的矩形，大家从中体会到了什么道理？

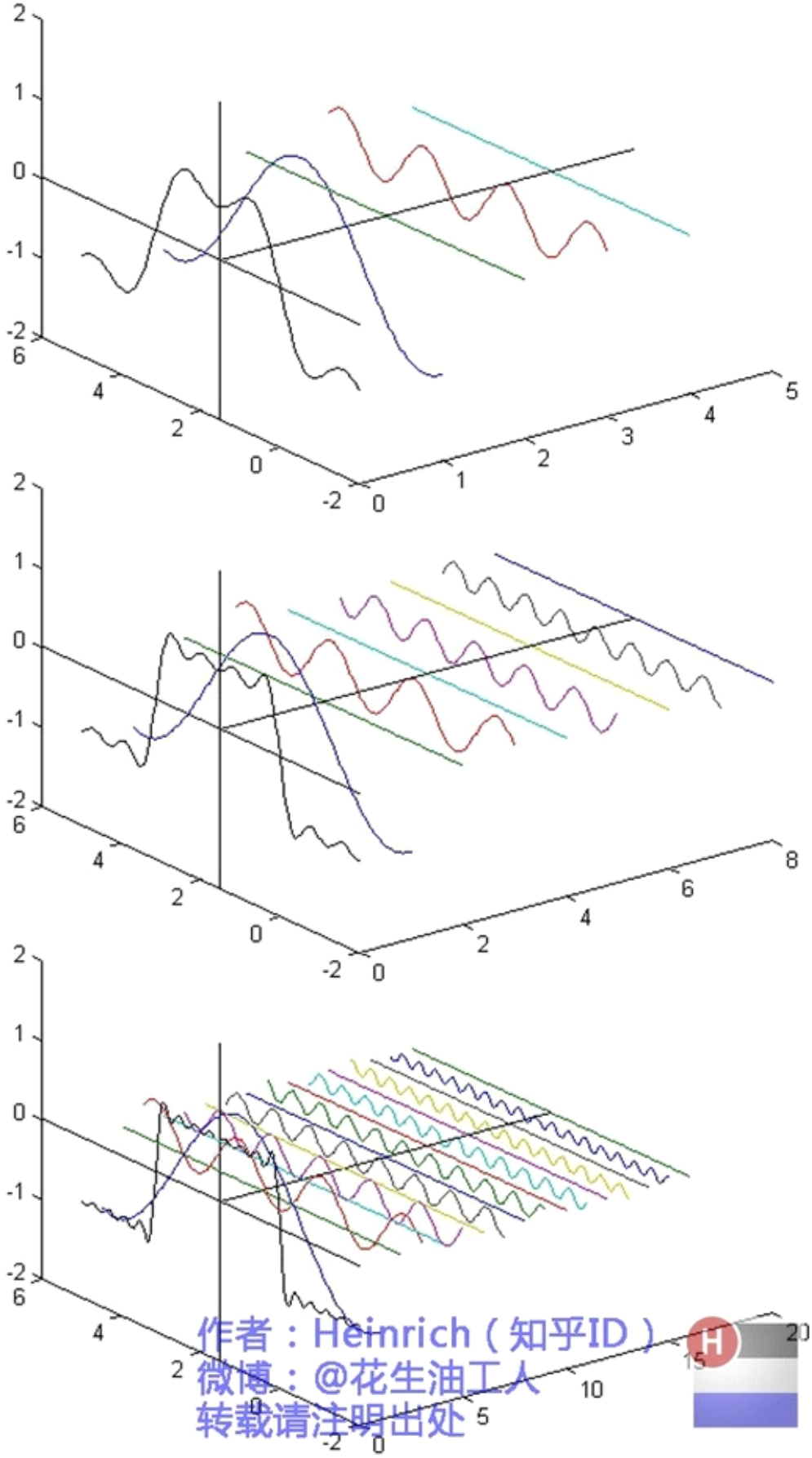
（只要努力，弯的都能掰直！）

随着叠加的递增，所有正弦波中上升的部分逐渐让原本缓慢增加的曲线不断变陡，而所有正弦波

中下降的部分又抵消了上升到最高处时继续上升的部分使其变为水平线。一个矩形就这么叠加而成了。但是要多少个正弦波叠加起来才能形成一个标准 90 度角的矩形波呢？不幸的告诉大家，答案是无穷多个。（上帝：我能让你们猜着我？）

不仅仅是矩形，你能想到的任何波形都是可以如此方法用正弦波叠加起来的。这是没有接触过傅里叶分析的人在直觉上的第一个难点，但是一旦接受了这样的设定，游戏就开始有意思起来了。

还是上图的正弦波累加成矩形波，我们换一个角度来看看：



在这几幅图中，最前面黑色的线就是所有正弦波叠加而成的总和，也就是越来越接近矩形波的那个图形。而后面依不同颜色排列而成的正弦波就是组合为矩形波的各个分量。这些正弦波按照频率从低到高从前向后排列开来，而每一个波的振幅都是不同的。一定有细心的读者发现了，每两个正弦波之间都还有一条直线，那并不是分割线，而是振幅为 0 的正弦波！也就是说，为了组成

特殊的曲线，有些正弦波成分是不需要的。

这里，不同频率的正弦波我们成为频率分量。

好了，关键的地方来了！！

如果我们把第一个频率最低的频率分量看作“1”，我们就有了构建频域的最基本单元。

对于我们最常见的有理数轴，数字“1”就是有理数轴的基本单元。

（好吧，数学称法为——基。在那个年代，这个字还没有其他奇怪的解释，后面还有正交基这样的词汇我会说吗?）

时域的基本单元就是“1 秒”，如果我们将一个角频率为

ω_1

的正弦波 cos（

ω_1

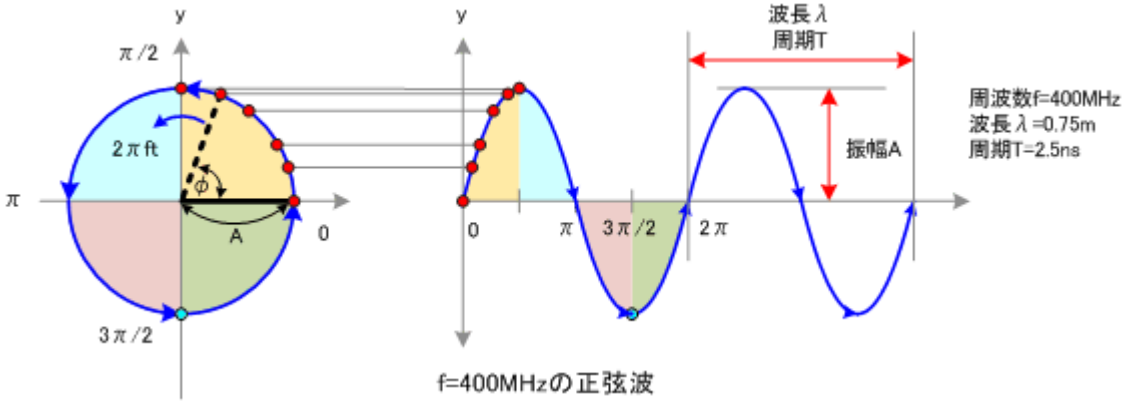
t）看作基础，那么频域的基本单元就是

ω_1

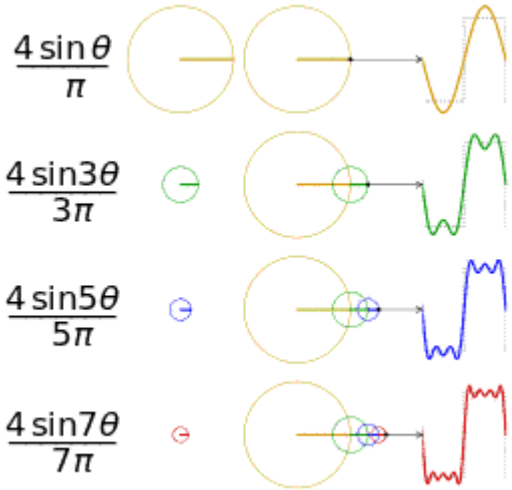
。

有了“1”，还要有“0”才能构成世界，那么频域的“0”是什么呢？cos（0t）就是一个周期无限长的正弦波，也就是一条直线！所以在频域，0 频率也被称为直流分量，在傅里叶级数的叠加中，它仅仅影响全部波形相对于数轴整体向上或是向下而不改变波的形状。

接下来，让我们回到初中，回忆一下已经死去的八戒，啊不，已经死去的老师是怎么定义正弦波的吧。

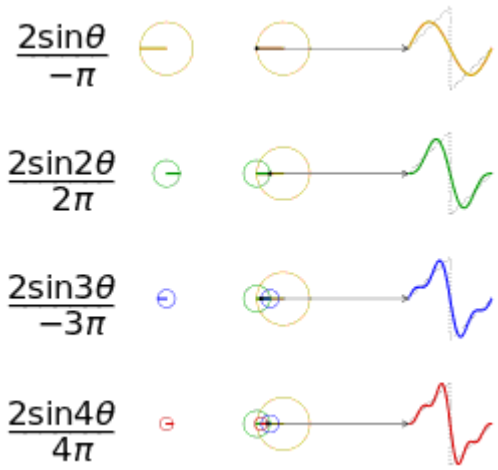


正弦波就是一个圆周运动在一条直线上的投影。所以频域的基本单元也可以理解为一个始终在旋转的圆



想看动图的同学请戳这里：

File:Fourier series square wave circles animation.gif

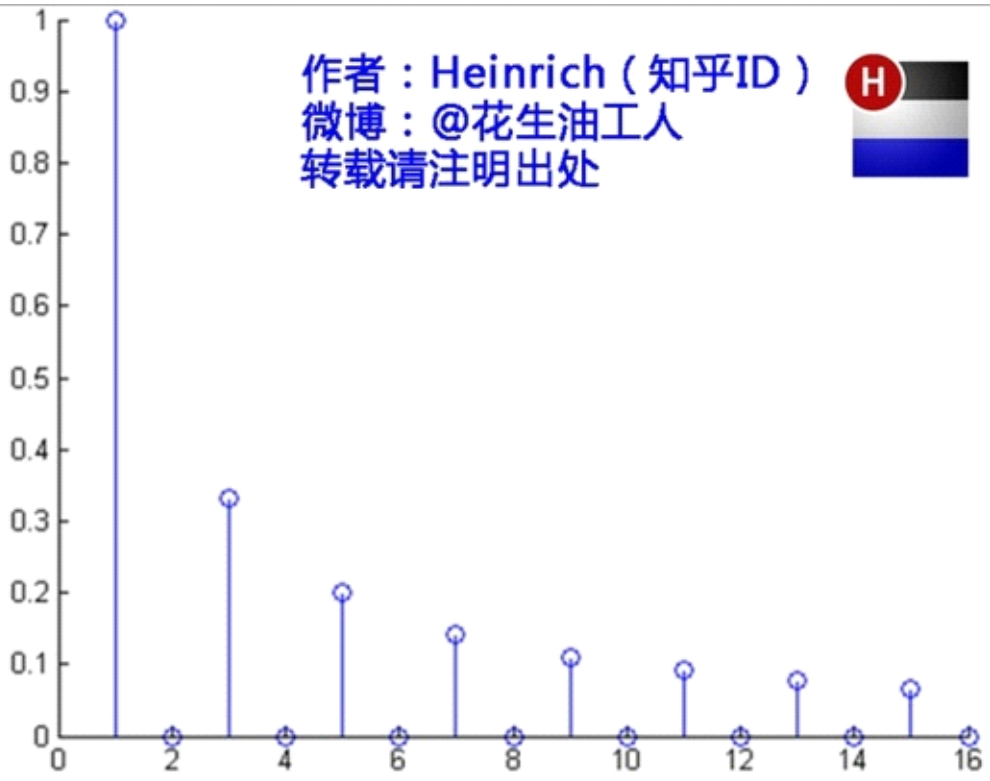


以及这里：

File:Fourier series sawtooth wave circles animation.gif

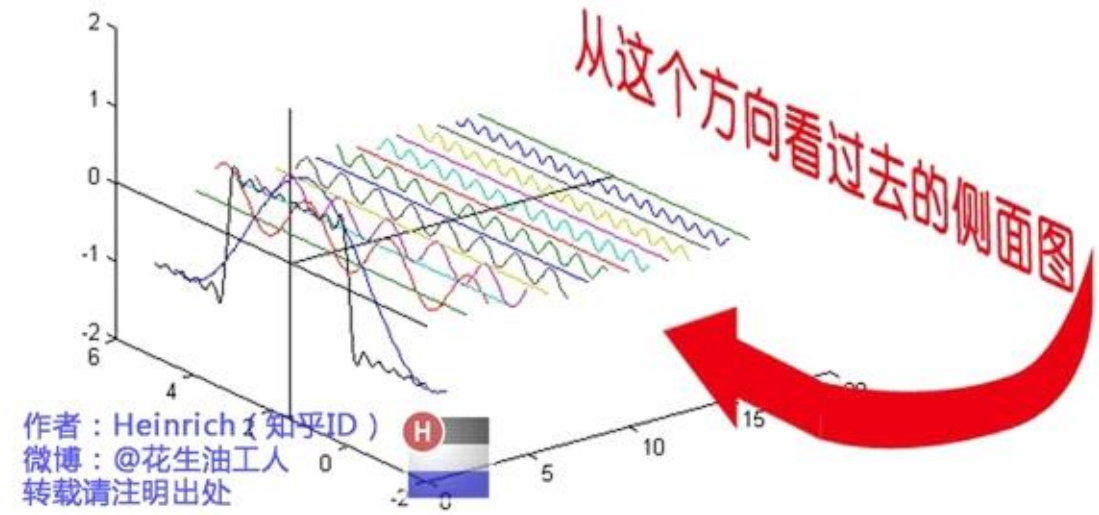
点出去的朋友不要被 wiki 拐跑了，wiki 写的哪有这里的文章这么没节操是不是。

介绍完了频域的基本组成单元，我们就可以看一看一个矩形波，在频域里的另一个模样了：

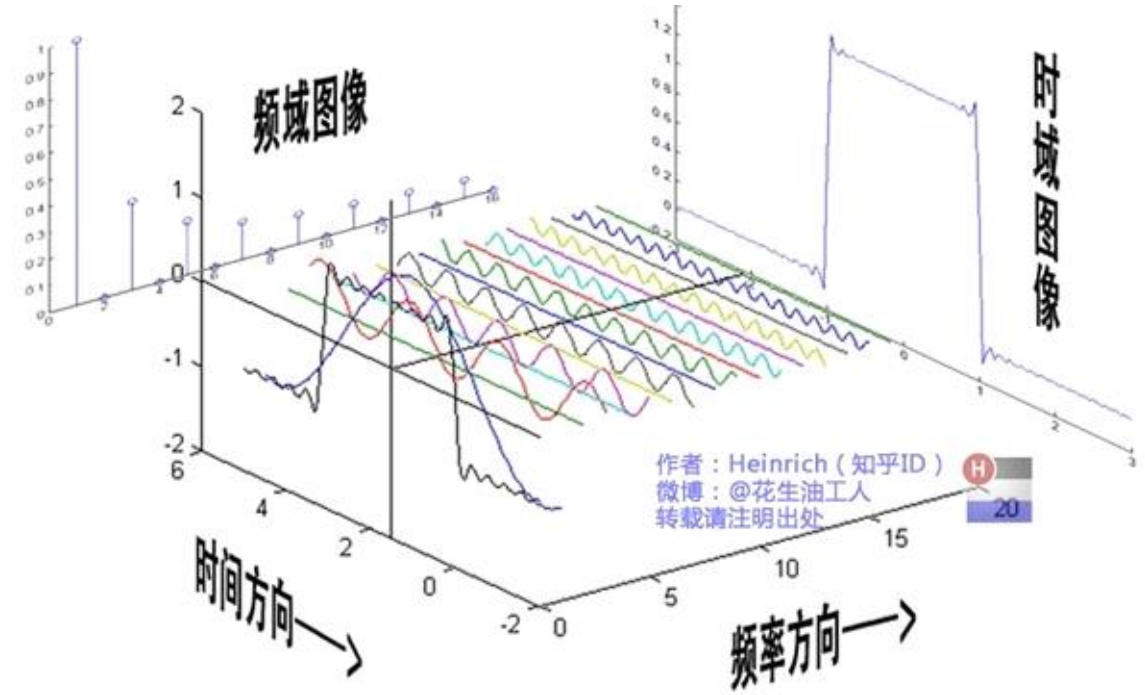


这是什么奇怪的东西？

这就是矩形波在频域的样子，是不是完全认不出来了？教科书一般就给到这里然后留给了读者无穷的遐想，以及无穷的吐槽，其实教科书只要补一张图就足够了：频域图像，也就是俗称的频谱，就是——



再清楚一点：



可以发现，在频谱中，偶数项的振幅都是0，也就对应了图中的彩色直线。振幅为 0 的正弦波。

动图请戳：

File:Fourier series and transform.gif

老实说，在我学傅里叶变换时，维基的这个图还没有出现，那时我就想到了这种表达方法，而

且，后面还会加入维基没有表示出来的另一个谱——相位谱。

但是在讲相位谱之前，我们先回顾一下刚刚的这个例子究竟意味着什么。记得前面说过的那句“世界是静止的”吗？估计好多人对这句话都已经吐槽半天了。想象一下，世界上每一个看似混乱的表象，实际都是一条时间轴上不规则的曲线，但实际这些曲线都是由这些无穷无尽的正弦波组成。我们看似不规律的事情反而是规律的正弦波在时域上的投影，而正弦波又是一个旋转的圆在直线上的投影。那么你的脑海中会产生一个什么画面呢？

我们眼中的世界就像皮影戏的大幕布，幕布的后面有无数的齿轮，大齿轮带动小齿轮，小齿轮再带动更小的。在最外面的小齿轮上有一个小人——那就是我们自己。我们只看到这个小人毫无规律的在幕布前表演，却无法预测他下一步会去哪。而幕布后面的齿轮却永远一直那样不停的旋转，永不停歇。这样说来有些宿命论的感觉。说实话，这种对人生的描绘是我一个朋友在我们都是高中生的时候感叹的，当时想想似懂非懂，直到有一天我学到了傅里叶级数……

三、傅里叶级数（Fourier Series）的相位谱

上一章的关键词是：从侧面看。这一章的关键词是：从下面看。

在这一章最开始，我想先回答很多人的一个问题：傅里叶分析究竟是干什么用的？这段相对比较枯燥，已经知道了的同学可以直接跳到下一个分割线。

先说一个最直接的用途。无论听广播还是看电视，我们一定对一个词不陌生——频道。频道频道，就是频率的通道，不同的频道就是将不同的频率作为一个通道来进行信息传输。下面大家尝试一件事：

先在纸上画一个sin（x），不一定标准，意思差不多就行。不是很难吧。

好，接下去画一个sin（3x）+sin（5x）的图形。

别说标准不标准了，曲线什么时候上升什么时候下降你都不一定画的对吧？

好，画不出来不要紧，我把sin（3x）+sin（5x）的曲线给你，但是前提是你不知道这个曲线的方程式，现在需要你把sin（5x）给我从图里拿出去，看看剩下的是什么。这基本是不可能做到的。

但是在频域呢？则简单的很，无非就是几条竖线而已。

所以很多在时域看似不可能做到的数学操作，在频域相反很容易。这就是需要傅里叶变换的地方。尤其是从某条曲线中去除一些特定的频率成分，这在工程上称为滤波，是信号处理最重要的概念之一，只有在频域才能轻松的做到。

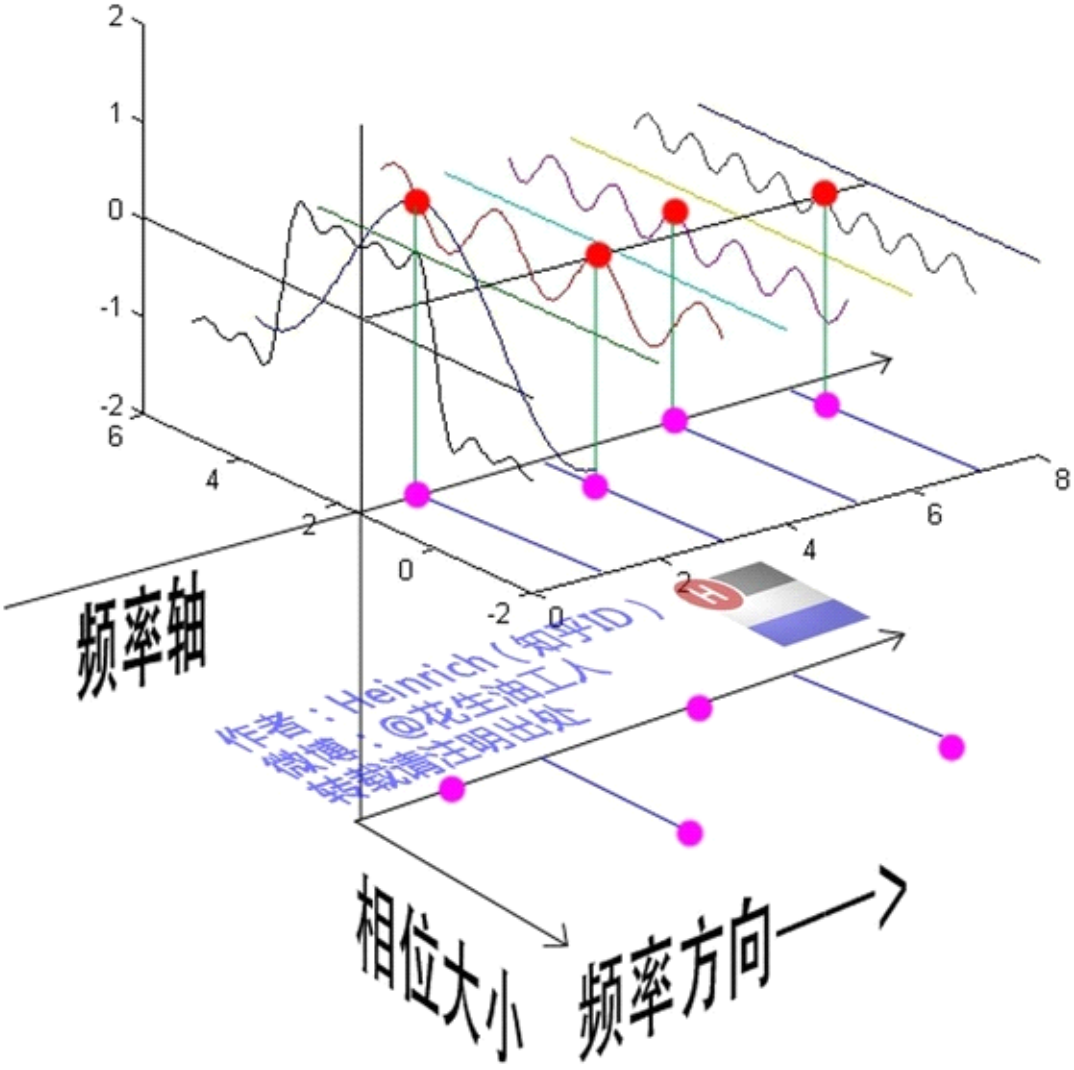
再说一个更重要，但是稍微复杂一点的用途——求解微分方程。（这段有点难度，看不懂的可以直接跳过这段）微分方程的重要性不用我过多介绍了。各行各业都用的到。但是求解微分方程却是一件相当麻烦的事情。因为除了要计算加减乘除，还要计算微分积分。而傅里叶变换则可以让微分和积分在频域中变为乘法和除法，大学数学瞬间变小学算术有没有。

傅里叶分析当然还有其他更重要的用途，我们随着讲随着提。

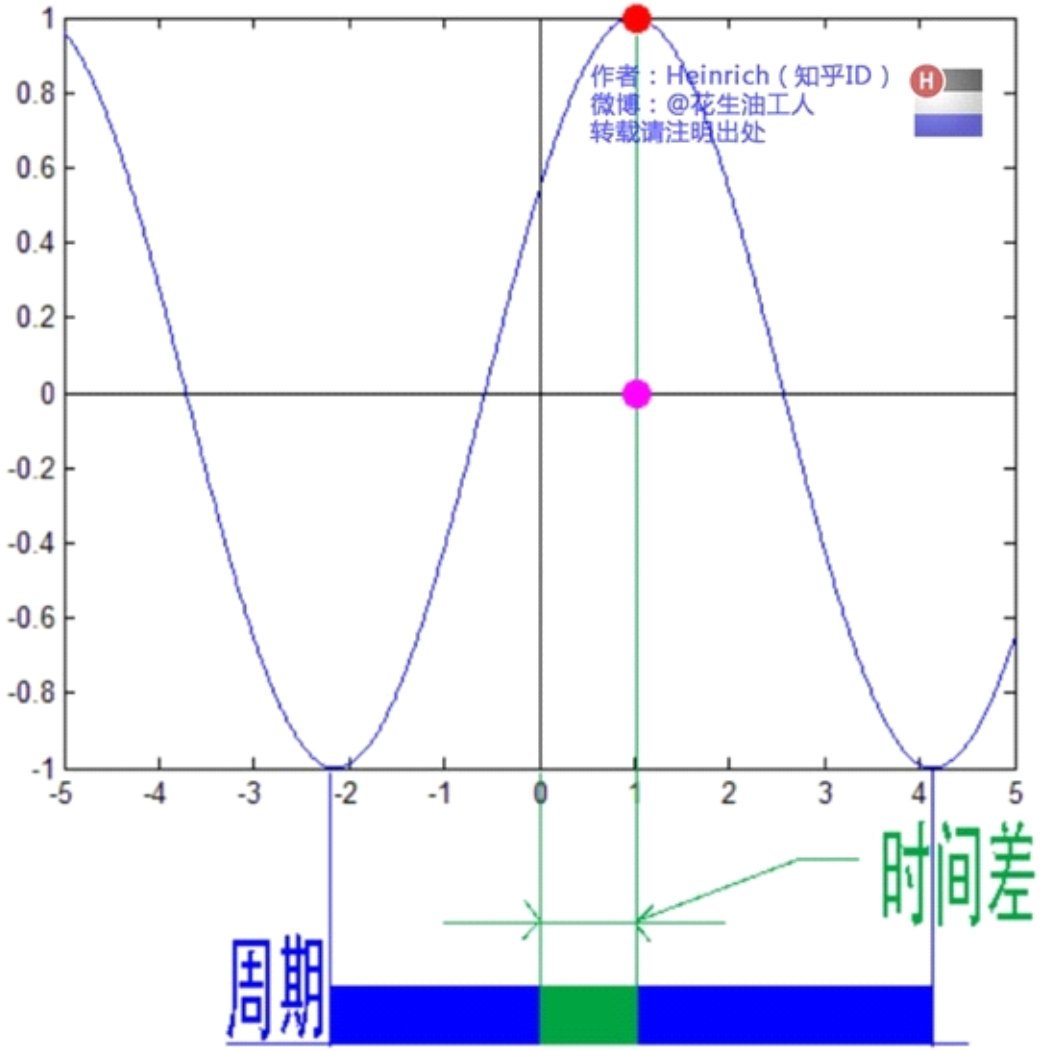
——

下面我们继续说相位谱：

通过时域到频域的变换，我们得到了一个从侧面看的频谱，但是这个频谱并没有包含时域中全部的信息。因为频谱只代表每一个对应的正弦波的振幅是多少，而没有提到相位。基础的正弦波A.sin(wt+θ)中，振幅，频率，相位缺一不可，不同相位决定了波的位置，所以对于频域分析，仅仅有频谱（振幅谱）是不够的，我们还需要一个相位谱。那么这个相位谱在哪呢？我们看下图，这次为了避免图片太混论，我们用7个波叠加的图。



鉴于正弦波是周期的，我们需要设定一个用来标记正弦波位置的东西。在图中就是那些小红点。小红点是距离频率轴最近的波峰，而这个波峰所处的位置离频率轴有多远呢？为了看的更清楚，我们将红色的点投影到下平面，投影点我们用粉色点来表示。当然，这些粉色的点只标注了波峰距离频率轴的距离，并不是相位。

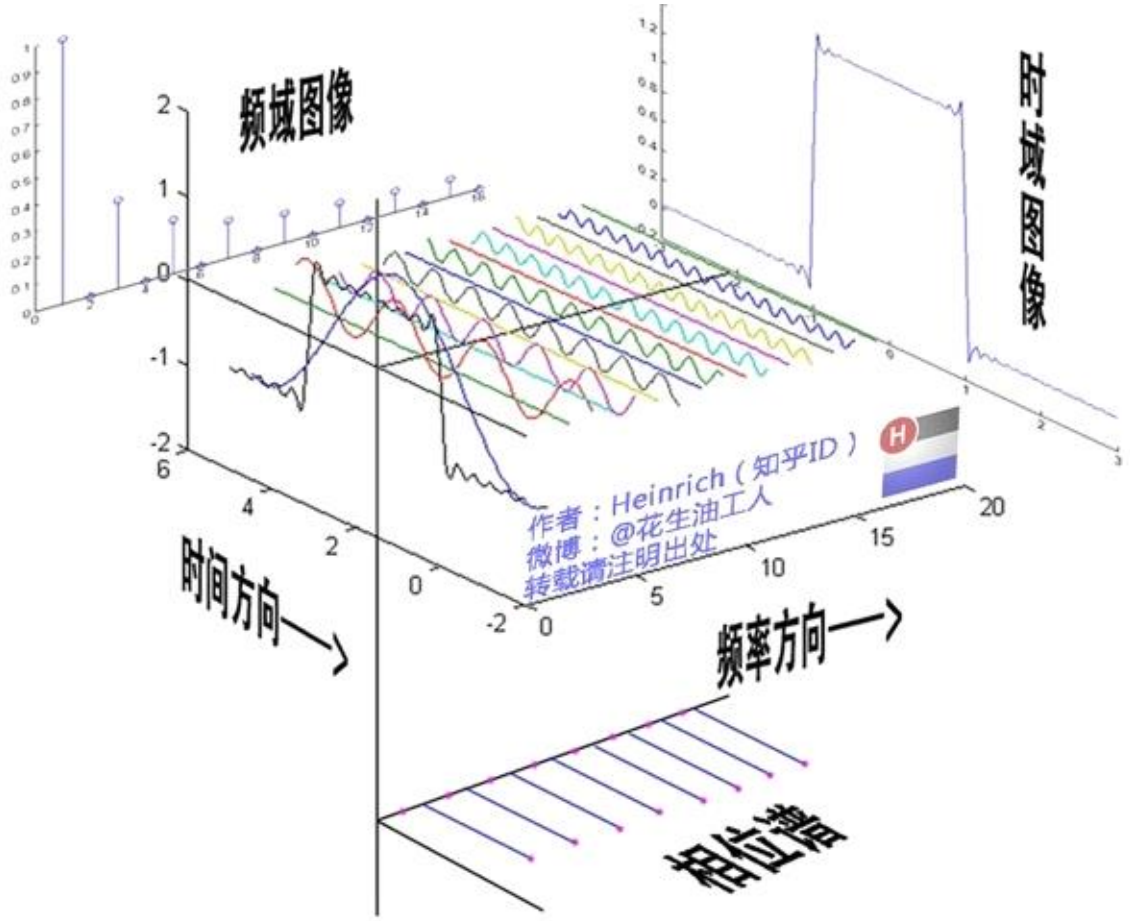


这里需要纠正一个概念：时间差并不是相位差。如果将全部周期看作2Pi或者360度的话，相位差则是时间差在一个周期中所占的比例。我们将时间差除周期再乘2Pi，就得到了相位差。

在完整的立体图中，我们将投影得到的时间差依次除以所在频率的周期，就得到了最下面的相位谱。所以，频谱是从侧面看，相位谱是从下面看。下次偷看女生裙底被发现的话，可以告诉她：“对不起，我只是想看看你的相位谱。”

注意到，相位谱中的相位除了0，就是Pi。因为cos（t+Pi）=-cos（t），所以实际上相位为Pi的波只是上下翻转了而已。对于周期方波的傅里叶级数，这样的相位谱已经是很简单的了。另外值得注意的是，由于cos（t+2Pi）=cos（t），所以相位差是周期的，pi和3pi，5pi，7pi都是相同的相位。人为定义相位谱的值域为(-pi，pi]，所以图中的相位差均为Pi。

最后来一张大集合：



四、傅里叶变换（Fourier Tranformation）

相信通过前面三章，大家对频域以及傅里叶级数都有了一个全新的认识。但是文章在一开始关于钢琴琴谱的例子我曾说过，这个栗子是一个公式错误，但是概念典型的例子。所谓的公式错误在哪里呢？

傅里叶级数的本质是将一个周期的信号分解成无限多分开的（离散的）正弦波，但是宇宙似乎并不是周期的。曾经在学数字信号处理的时候写过一首打油诗：

往昔连续非周期，
回忆周期不连续，
任你ZT、DFT，
还原不回去。

（请无视我渣一样的文学水平……）

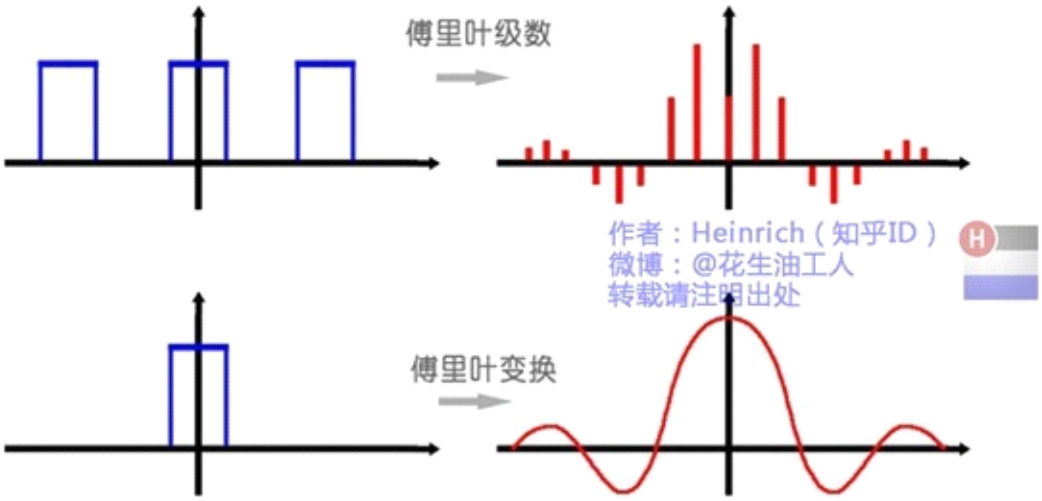
在这个世界上，有的事情一期一会，永不再来，并且时间始终不曾停息地将那些刻骨铭心的往昔连续的标记在时间点上。但是这些事情往往又成为了我们格外宝贵的回忆，在我们大脑里隔一段时间就会周期性的蹦出来一下，可惜这些回忆都是零散的片段，往往只有最幸福的回忆，而平淡的回忆则逐渐被我们忘却。因为，往昔是一个连续的非周期信号，而回忆是一个周期离散信号。

是否有一种数学工具将连续非周期信号变换为周期离散信号呢？抱歉，真没有。

比如傅里叶级数，在时域是一个周期且连续的函数，而在频域是一个非周期离散的函数。这句话比较绕嘴，实在看着费事可以干脆回忆第一章的图片。

而在我们接下去要讲的傅里叶变换，则是将一个时域非周期的连续信号，转换为一个在频域非周期的连续信号。

算了，还是上一张图方便大家理解吧：



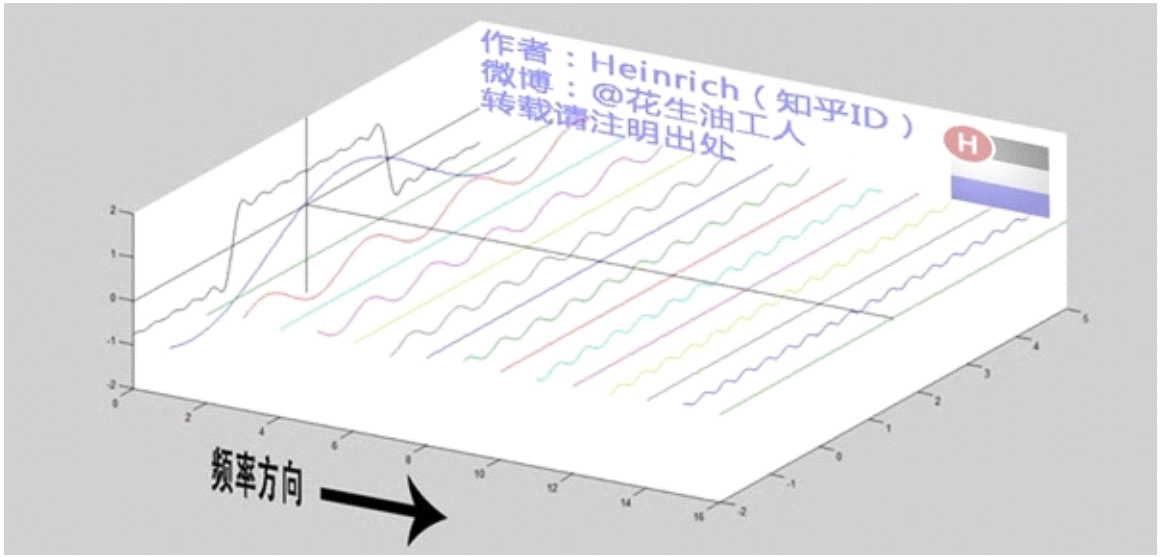
或者我们也可以换一个角度理解：傅里叶变换实际上是对一个周期无限大的函数进行傅里叶变换。

所以说，钢琴谱其实并非一个连续的频谱，而是很多在时间上离散的频率，但是这样的一个贴切的比喻真的是很难找出第二个来了。

因此在傅里叶变换在频域上就从离散谱变成了连续谱。那么连续谱是什么样子呢？

你见过大海么？

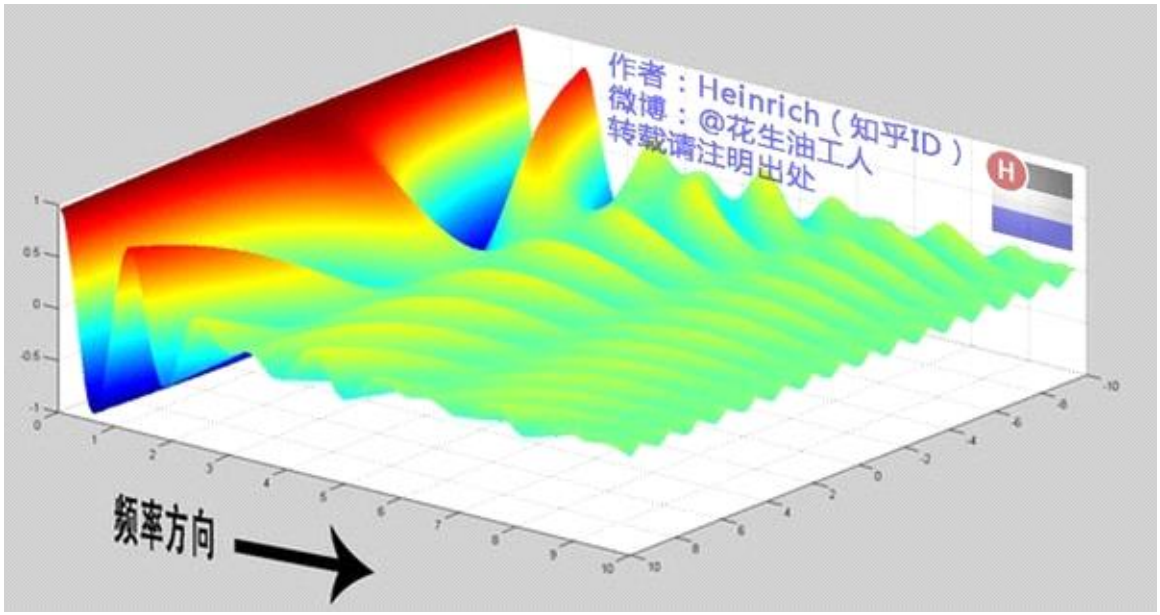
为了方便大家对比，我们这次从另一个角度来看频谱，还是傅里叶级数中用到最多的那幅图，我们从频率较高的方向看。



以上是离散谱，那么连续谱是什么样子呢？

尽情的发挥你的想象，想象这些离散的正弦波离得越来越近，逐渐变得连续……

直到变得像波涛起伏的大海：



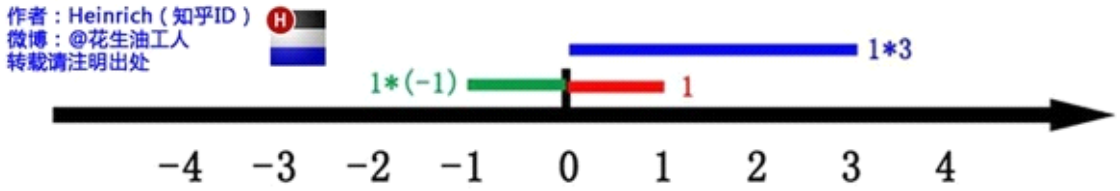
很抱歉，为了能让这些波浪更清晰的看到，我没有选用正确的计算参数，而是选择了一些让图片更美观的参数，不然这图看起来就像屎一样了。

不过通过这样两幅图去比较，大家应该可以理解如何从离散谱变成了连续谱的了吧？原来离散谱的叠加，变成了连续谱的累积。所以在计算上也从求和符号变成了积分符号。

不过，这个故事还没有讲完，接下去，我保证让你看到一幅比上图更美丽壮观的图片，但是这里需要介绍到一个数学工具才能然故事继续，这个工具就是——

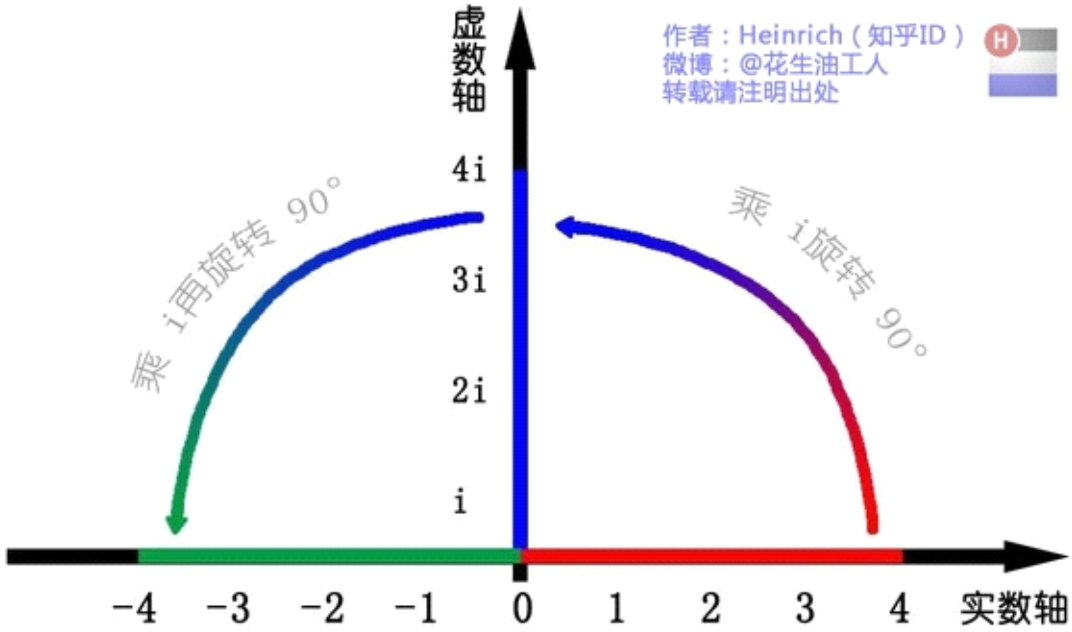
五、宇宙要帅第一公式：欧拉公式

虚数i这个概念大家在高中就接触过，但那时我们只知道它是-1 的平方根，可是它真正的意义是什么呢?



这里有一条数轴，在数轴上有一个红色的线段，它的长度是1。当它乘以 3 的时候，它的长度发生了变化，变成了蓝色的线段，而当它乘以-1 的时候，就变成了绿色的线段，或者说线段在数轴上围绕原点旋转了 180 度。

我们知道乘-1 其实就是乘了两次 i 使线段旋转了 180 度，那么乘一次 i 呢——答案很简单——旋转了 90 度。



同时，我们获得了一个垂直的虚数轴。实数轴与虚数轴共同构成了一个复数的平面，也称复平面。这样我们就了解到，乘虚数i的一个功能——旋转。

现在，就有请宇宙第一耍帅公式欧拉公式隆重登场——

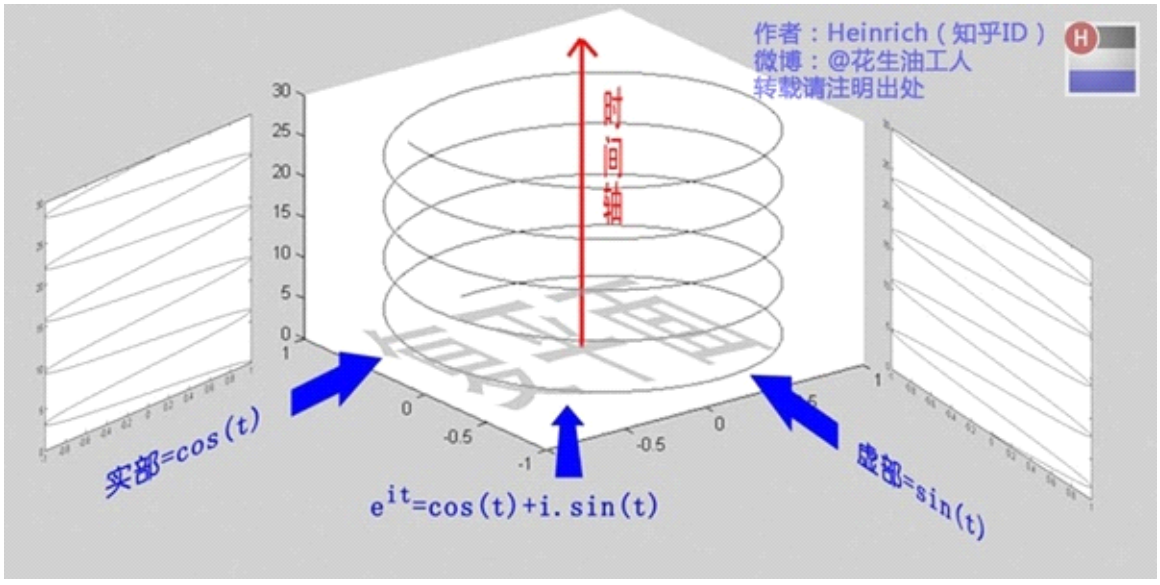
$$e^{ix} = \cos x + i\sin x$$

这个公式在数学领域的意义要远大于傅里叶分析，但是乘它为宇宙第一耍帅公式是因为它的特殊形式——当x等于 Pi 的时候。

$$e^{i\pi} + 1 = 0$$

经常有理工科的学生为了跟妹子表现自己的学术功底，用这个公式来给妹子解释数学之美：”石榴姐你看，这个公式里既有自然底数e，自然数 1 和0，虚数i还有圆周率 pi，它是这么简洁，这么美丽啊！“但是姑娘们心里往往只有一句话：”臭屌丝……“

这个公式关键的作用，是将正弦波统一成了简单的指数形式。我们来看看图像上的涵义：



欧拉公式所描绘的，是一个随着时间变化，在复平面上做圆周运动的点，随着时间的改变，在时间轴上就成了一条螺旋线。如果只看它的实数部分，也就是螺旋线在左侧的投影，就是一个最基础的余弦函数。而右侧的投影则是一个正弦函数。

关于复数更深的理解，大家可以参考：

[复数的物理意义是什么？]

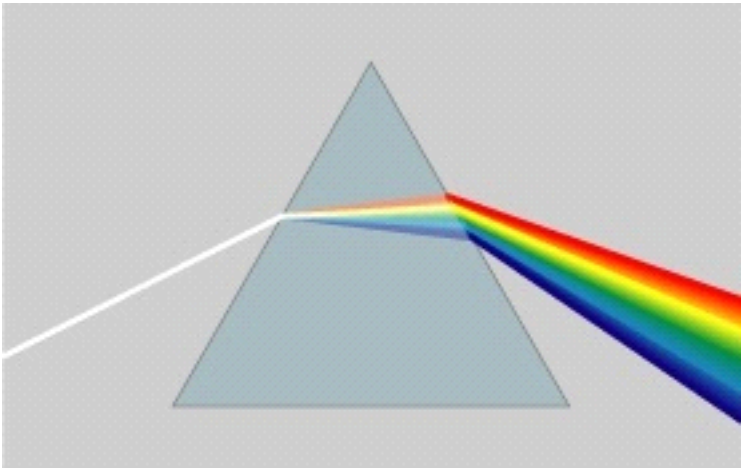
这里不需要讲的太复杂，足够让大家理解后面的内容就可以了。

六、指数形式的傅里叶变换

有了欧拉公式的帮助，我们便知道：正弦波的叠加，也可以理解为螺旋线的叠加在实数空间的投影。而螺旋线的叠加如果用一个形象的栗子来理解是什么呢？

光波

高中时我们就学过，自然光是由不同颜色的光叠加而成的，而最著名的实验就是牛顿师傅的三棱镜实验：



所以其实我们在很早就接触到了光的频谱，只是并没有了解频谱更重要的意义。

但不同的是，傅里叶变换出来的频谱不仅仅是可见光这样频率范围有限的叠加，而是频率从 0 到无穷所有频率的组合。

这里，我们可以用两种方法来理解正弦波：

第一种前面已经讲过了，就是螺旋线在实轴的投影。

另一种需要借助欧拉公式的另一种形式去理解：

$$e^{it} = cos(t) + i.sin(t)$$

$$e^{-it} = cos(t) - i.sin(t)$$

将以上两式相加再除2，得到：

$$cos(t) = \frac{e^{it} + e^{-it}}{2}$$

这个式子可以怎么理解呢？

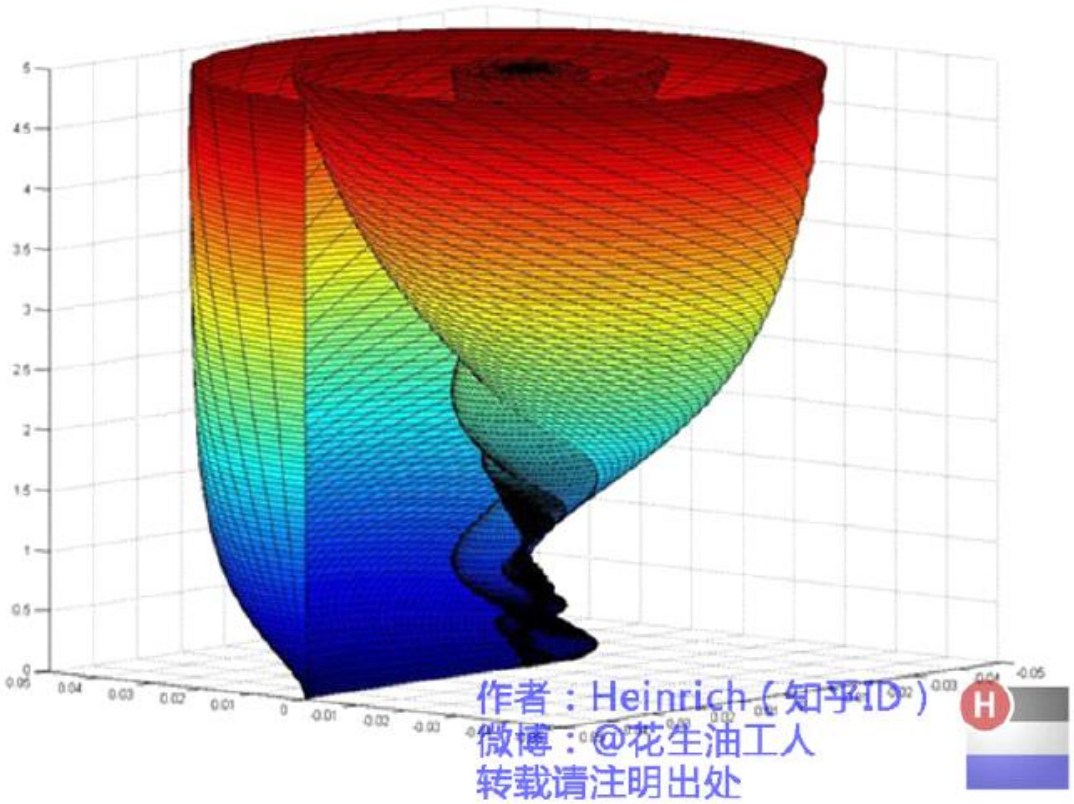
我们刚才讲过，e^(it)可以理解为一条逆时针旋转的螺旋线，那么e^(-it)则可以理解为一条顺时针旋转的螺旋线。而 cos (t)则是这两条旋转方向不同的螺旋线叠加的一半，因为这两条螺旋线的虚数部分相互抵消掉了！

举个例子的话，就是极化方向不同的两束光波，磁场抵消，电场加倍。

这里，逆时针旋转的我们称为正频率，而顺时针旋转的我们称为负频率（注意不是复频率）。

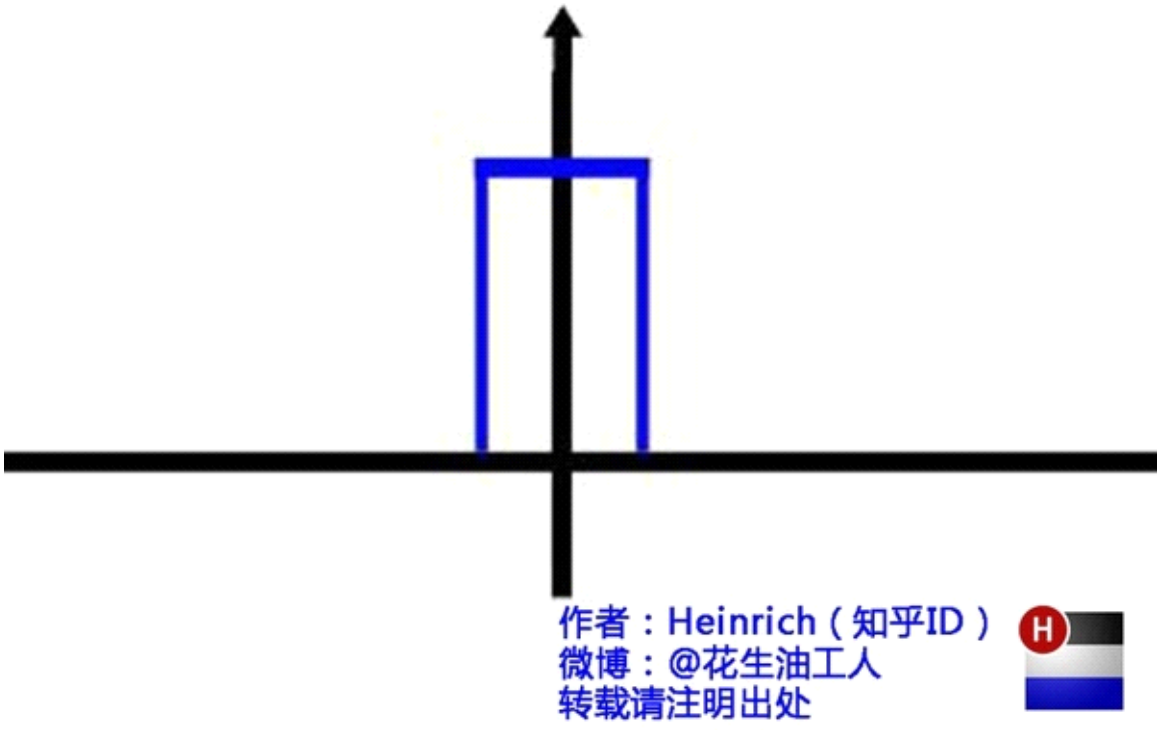
好了，刚才我们已经看到了大海——连续的傅里叶变换频谱，现在想一想，连续的螺旋线会是什么样子：

想象一下再往下翻：



是不是很漂亮？

你猜猜，这个图形在时域是什么样子？

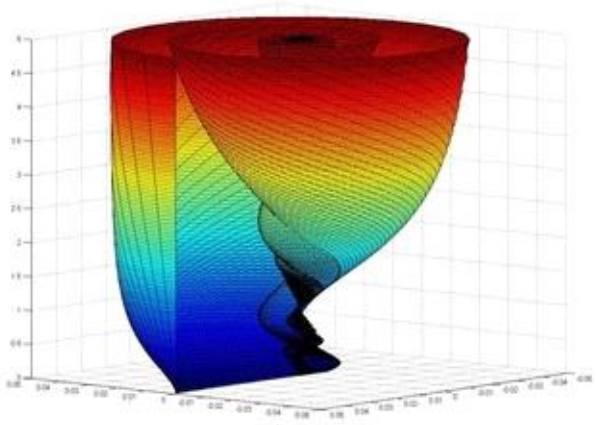


哈哈，是不是觉得被狠狠扇了一个耳光。数学就是这么一个把简单的问题搞得很复杂的东西。

顺便说一句，那个像大海螺一样的图，为了方便观看，我仅仅展示了其中正频率的部分，负频率的部分没有显示出来。

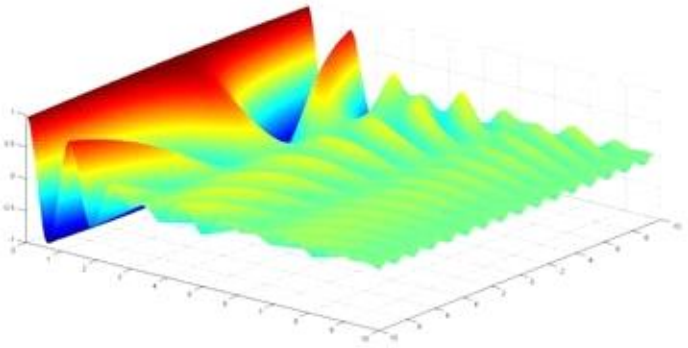
如果你认真去看，海螺图上的每一条螺旋线都是可以清楚的看到的，每一条螺旋线都有着不同的振幅（旋转半径），频率（旋转周期）以及相位。而将所有螺旋线连成平面，就是这幅海螺图了。

好了，讲到这里，相信大家对傅里叶变换以及傅里叶级数都有了一个形象的理解了，我们最后用一张图来总结一下：



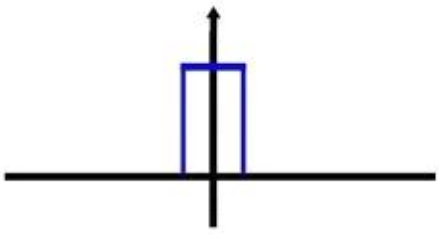
复频域

投影到实数空间



频域

各频率成分累积



时域

作者：Heinrich（知乎ID）
微博：@花生油工人
转载请注明出处

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5071677>

北航第十一届程序设计竞赛网络预赛题解

2016年5月6日　　13:32

A. 模式

计算机学院是6系，软件学院是21系，对于输入的学号只需要判断⌊id10000⌋是多少即可。

B. 并联变阻器

题意即统计满足a,b≤N且aba+b是整数的二元组(a,b)的个数，条件也即a,b≤N且(a+b)|ab。

令r=gcd(a,b)，则有gcd(ar,br)=1，再令a=rx,b=ry，那么(a+b)|ab 可以表示为r(x+y)|r2xy，也就是(x+y)|rxy。

由于x与y互质，所以gcd(x+y,x)=gcd(x+y,y)=gcd(x,y)=1，(x+y)与x,y也是互质的，则必然有(x+y)|r。

令r=k(x+y)，则a=kx(x+y),b=ky(x+y)，可以发现a,b≤N对应着x,y≤N−−√，只需要枚举不超过N−−√的互质数对(x,y)，即可计算出对应的每一组解，这样做的时间复杂度是O(N−−√2)=O(N)的，其中求最大公约数的部分可以通过预处理达到O(1)。

注意到本题的数据组数较大，单组数据使用O(N)算法也会超时，但是枚举所有解的时候也确定了这组解是属于N≥max(a,b)的所有N，所以将这组(a,b)统计到对应的max(a,b)，再求一遍前缀和即可得到所有答案，预处理复杂度O(N)，单点查询O(1)。

C. 抽奖箱

题意可以稍作简化，考虑成n次抽奖，每次只有m位老师和ai位学生，需要求抽到k位老师时，在这k位学生之前的同学平均情况下有多少位。不同班的学生之间互相不影响。

注意到样例解释里不同情况下的概率是不一样的，不便于分析，可以将抽取的序列补全成抽完(m+ai)个结果的一个排列，这样每个排列的概率都是1(m+ai)!。

一个直观的结论是，可以认为学生是等概率分布在老师之间的，即任意两个相邻的老师之间平均情况下有相同数量的学生，那么m位老师将序列划分成(m+1)个段，每段平均情况下有aim+1个学生，所求即前k个段里平均情况下的学生个数，也就是ai·km+1。如果理解了数学期望的线性性，应该会很快得到这个结论。详细的证明可以参考超几何分布的期望。

D. 最大公约数

最大公约数满足结合律，所以题目所说相等的gcd就是原数列的gcd，不妨设为d。

设f[i]为前i个数能分的最大段数，枚举最后一个段是[j+1,i]，则有gcdik=j+1ak=d，而f[i]=maxf[j]+1。如果j不存在，可以认为f[i]是不合法的部分，令f[i]=0即可。

不难证明满足条件的j是一段前缀区间，而且f[i]是单调的，所以最大的f[j]一定是尽量靠右的，可以直接找到这个j来对f[i]更新答案。

区间gcd可以预处理ST表得到，或者顺着推以每个点结尾的区间gcd即可，可以证明以每个点结尾的区间gcd的值个数是不超过logn个的，所以整体的复杂度是O(nlog2n)。

E. 矩阵

设 Xi代表第i行所减少的权值，设 Yj代表第j 列所增加的权值。

则 Cij=Yj−Xi, 推出 Xi+Cij≤Yj 和 Yj−Cij≤Xi，根据差分关系建边。原问题就变成：是否所有的环权值都为0。n+m 只有 100，所以floyd和spfa找环什么的都可以过。

PS:没想到大部分人都用高斯消元过掉了。

F. 序列

对于一个固定的排列p，权值为 1+∑n−1i=1[pi>pi+1]。所以相邻两个数字，如果前面数字大于后面数字则对答案贡献1。

公式：(n2)(n−11)·(n−2)!+n!=(n+1)!2。

G. 就是这么巧

这题的结论： 1.对于符合条件的(a,b)，a2+b2ab+1是一个完全平方数

2.对于给定的n,a2+b2ab+1=n2的所有解是数列a1=n,a2=n3,ai=n2ai−1−ai−2,i>2中的相邻两项。

证明：

- 结论1：保加利亚选手的证明
- 结论2：参考结论1的证明

令：a2+b2ab+1=k>0，a≥b，并移项化简得:a2−kb·a+b2−k=0。

对于给定的k，我们将a,b的范围从正整数扩大为整数。那么a,b不会异号（否则a2−kb·a+b2−k≥a2+k+b2−k>0，矛盾），即ab≥0。

已知a是一元二次方程x2−kb·x+b2−k=0的一个解，设另一个解为a′，根据韦达定理:

a+a′=kb,aa′=b2−k=>a′=b2−ka<b≤a

公式a+a′=kb就是An−1+An+1=kAn。

就是说任意给定一组解(b,a)，我们可以求出另一组解(a′,b)，且a′<b≤a，继续迭代可以得到一个无穷数列{Ai},(Ai,Ai+1)都是解。

当k>1时，这个数列是递增的；并且关于原点对称，因为(−a,−b)也是一组是解；并且0是数列中的某一项（否则必然存在Ai<0<Ai+1）；从而对于给定的k，所有的解都在这个数列中。

H. 高中数学题

题目给出了一个仅含an、Sn的公式，根据高中数学知识，很容易解出它的通项：an=2n+1

我们要求的是这个序列的前n项异或和

对于自然数的前n项异或和Xn，有如下规律：

Xn={|||||n,1,n+1,0,n≡0mod4n≡1mod4n≡2mod4n≡3mod4

本题中的序列就是自然数序列左移一位，再加1得到的

因此先将自然数序列的前n项异或和算出，之后将其左移一位再单独考虑最后一位的情况就好了。

I. 神奇宝贝大师

首先很重要的一点，我们可以发现X方向位置的选择和Y方向位置的选择是相互独立的。如果能想到这点这题就非常非常简单了。

假设我们最后答案是(ansx, ansy)，我们只要独立的找到ansx和ansy就好了。 而只考虑X方向或者Y方向的话，就把原题转换成了一维的问题：给一个数组a，找一个位置x使得∑ai·|i−x|最小。

而这个一维的问题就很好解决了。可以O(n)扫一遍维护一下前后的距离暴力统计取min，或者直接找中间位置。而这个题更加简单一些，2000的数据范围直接O(n2)暴力枚举都是可以的。

J. 铅导体

这个题目的操作是在原图的每条边上加上一个dt，我们可以发现，最终影响答案的是每条路径的

边数。我们不妨把所有从S到T的路径表示为A+B·dt的形式，A代表原图该路径的长度，B代表该路径的边数。

而A+B·dt是一束直线，且我们显然可以将其优化为n条直线(对每个B，取最小的A)。那么对于每个询问dt，答案就是这一束直线在x=dt处的最小值。我们可以处理出一个最多由n条边组成的下凸壳，对每个询问二分查找其所在的凸壳上的段，即可直接求出答案。

由于前面求n条直线的复杂度为O(nm),求下凸壳的复杂度为O(n),回答询问的复杂度为O(qlogn),所以总的复杂度为O(nm+qlogn)。题目中的nm比较大，但是实际上在图中跑的速度还是很快的。另外，由于数据未进行特别的构造，导致暴力处理n条直线，并且询问时枚举n条直线计算最值的算法也通过了此题。

K. 危险密码

两个字符串的编辑距离即一个串通过添加、删除、修改三种操作变成另外一个串的最少操作次数。

对于一个字符串s，设它的长度为n，可以发现h=(∑n−1i=0ai·Kn−1−i)modM，枚举添加、删除、修改的一个字符串，计算新串的h′即可。

对于修改第i(0≤i<n)个位置为c，h′=h+(c−ai)·Kn−1−i。

对于添加和删除需要一些额外的信息，令prei表示字符串s的前i个字符表示的h值，pren即s的h，再令sufi=pren−prei·Kn−i+1。

对于在第i(0≤i≤n)个字符前添加一个c，h′=prei·Kn−i+1+c·Kn−i+sufi+1。

对于删除第i(0≤i<n)个字符，h′=prei−1·Kn−i+sufi+1。

L. 偶回文串

题意即统计有多少个连续的子串满足在它里面出现的字符都出现了偶数次，满足这样条件的子串总能通过重排字符的顺序得到一个偶回文串。

任意一个子串里某个字符的出现次数可以被表示成两个前缀字符串里出现次数的差，例如abbababbabbab的子串ababba，就可以表示成abbababba和abb的差，如果这两个前缀串里任意一个字符出现的次数在模2意义下是相等的，那么他们的差对应的子串就是一个合法的解。以第i个字符结尾的前缀串和以第i+1个字符结尾的前缀串只差一个字符，可以通过线性递推得到所有的前缀串的26个字符出现次数的奇偶性，可以发现每个前缀串对应26个不是0就是1的数字，可以将其压缩成一个二进制数字si，si的第k位对应第k个字符出现次数的奇偶性，添加一个字符可以利用二进制不进位加法，其中二进制不进位加法可以用异或(xor)来表示。

算出所有的si之后，可以通过C++ STL map或手写散列函数统计相同的si出现个数，也可以直接进行排序将相同的si排到一起。不妨设有a个si是相等的，那么它们可以对应a·(a−1)2个子串，分别考虑每组相等的si，将贡献累加即可。时间复杂度O(nlogn)。

M. 我是鱼

一个数和自己异或(⊕)结果为0，所以如果只有一个数是奇数全部异或起来就能得到结果，如果有两个数是奇数，假设为a，b，把所有数异或起来的结果等于a⊕b，设c=a⊕b，则c≠0，c的二进制表示中必有某一位为1，假设为第x位，那么将所有第x位为0的异或起来，所有为1的异或起来就能得到a，b。

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5052737>

LeetCode 3 Longest Substring Without Repeating Characters（最长不重复子序列）

2016年5月6日 13:33

题目来源：https://leetcode.com/problems/longest-substring-without-repeating-characters/

Given a string, find the length of the longest substring without repeating characters. For example, the longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3. For "bbbbb" the longest substring is "b", with the length of 1

解题思路：

用一个start来记录目前字符串的开头
用exist[MAX]来记录目前字符串中出现过的字母
用pos[MAX]来记录出现过的字符串的字母的位置

然后我们往后走一位，然后和exist来比较看这个字母是否已经出现过了。

1 如果出现过了，那么我们把start移动到之前那个字母出现的位置的后一位，end往后移动一位
2 如果没有出现过，那么我们就把end往后移动一位
提交代码：

```
 1 class Solution {
 2 public:
 3     int lengthOfLongestSubstring(string s) {
 4         // Start typing your C/C++ solution below
 5         // DO NOT write int main() function
 6         int locs[256];//保存字符上一次出现的位置
 7         memset(locs, -1, sizeof(locs));
 8 
 9         int idx = -1, max = 0;//idx为当前子串的开始位置-1
10         for (int i = 0; i < s.size(); i++)
11         {
12             if (locs[s[i]] > idx)//如果当前字符出现过，那么当前子串的起始位置为这个字符上一次出现的位置+1
13             {
14                 idx = locs[s[i]];
15             }
16 
17             if (i - idx > max)
18             {
19                 max = i - idx;
20             }
21 
22             locs[s[i]] = i;
23         }
24         return max;
25     }
26 };
```

其他解题方法：

```
 1 #include <bits/stdc++.h>
 2 #define MAX 110
 3 
 4 using namespace std;
 5 
 6 int pos[MAX], exist[MAX];
 7 
 8 int main()
```

```
 9 {
10     string s;
11     int lens,i,j,start,max_num;
12     while(cin>>s)
13     {
14         lens=s.size();
15         max_num=0,start=0;
16         memset(pos,0,sizeof(pos));
17         memset(exist,0,sizeof(exist));
18         for(i=0;i<lens;i++)
19         {
20             if(exist[s[i]-'a'])
21             {
22                 for(j=start;j<=pos[s[i]-'a'];j++)
23                     exist[s[j]-'a']=0;
24                 start=pos[s[i]-'a']+1;
25                 exist[s[i]-'a']=1;
26                 pos[s[i]-'a']=i;
27             }
28             else
29             {
30                 exist[s[i]-'a']=1;
31                 pos[s[i]-'a']=i;
32                 max_num=max(max_num,i-start+1);
33             }
34         }
35         printf("%d\n",max_num);
36     }
37 }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5050220>

UVa 112 - Tree Summing（树的各路径求和,递归）

2016年5月6日　　13:33

题目来源：https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=3&page=show_problem&problem=48

Tree Summing

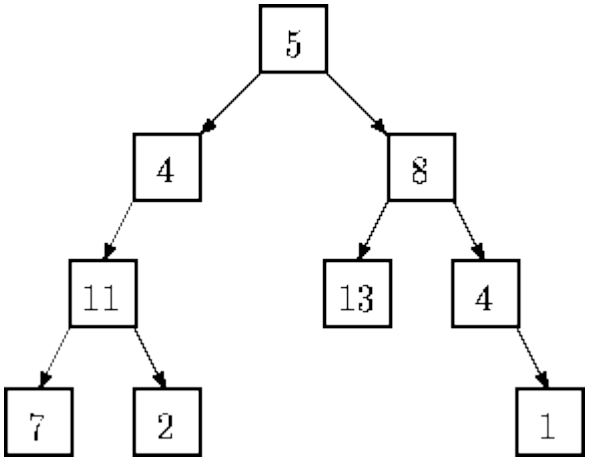
Background

LISP was one of the earliest high-level programming languages and, with FORTRAN, is one of the oldest languages currently being used. Lists, which are the fundamental data structures in LISP, can easily be adapted to represent other important data structures such as trees.

This problem deals with determining whether binary trees represented as LISP S-expressions possess a certain property.

The Problem

Given a binary tree of integers, you are to write a program that determines whether there exists a root-to-leaf path whose nodes sum to a specified integer. For example, in the tree shown below there are exactly four root-to-leaf paths. The sums of the paths are 27, 22, 26, and 18.



Binary trees are represented in the input file as LISP S-expressions having the following form.

empty tree ::= ()

tree ::= *empty tree* (integer *tree tree*)

The tree diagrammed above is represented by the expression (5 (4 (11 (7 () ()) (2 () ())) ()) (8 (13 () ()) (4 () (1 () ()))))

Note that with this formulation all leaves of a tree are of the form (integer () ())

Since an empty tree has no root-to-leaf paths, any query as to whether a path exists whose sum is a specified integer in an empty tree must be answered negatively.

The Input

The input consists of a sequence of test cases in the form of integer/tree pairs. Each test case consists of an integer followed by one or more spaces followed by a binary tree formatted as an S-expression as described above. All binary tree S-expressions will be valid, but expressions may be spread over several lines and may contain spaces. There will be one or more test cases in an input file, and input is terminated by end-of-file.

The Output

There should be one line of output for each test case (integer/tree pair) in the input file. For each pair *I*,*T* (*I* represents the integer, *T*represents the tree) the output is the string *yes* if there is a root-to-leaf path in *T* whose sum is *I* and *no* if there is no path in *T* whose sum is *I*.

Sample Input

```
22 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
20 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
10 (3
     (2 (4 () () )
        (8 () () ) )
     (1 (6 () () )
        (4 () () ) ) )
5 ()
```

Sample Output

```
yes
no
yes
no
```

解题思路:

题目给出树一种定义表达式.每组数据给出目标数据target,以及树的结构表达式.要求判断所给的树中是否存在一条路径满足其上节点的和等于target,如果存在输出yes,否则输出no.

所给的树属于二叉树,但是不一定是满二叉树,所以对于所给的树进行左右递归计算,如果存在路径满足则输出yes,否则输出no

推荐博客1:

http://www.cnblogs.com/devymex/archive/2010/08/10/1796854

.html

推荐博客2：
http://blog.csdn.net/zcube/article/details/8545544

推荐博客3：
http://blog.csdn.net/mobius_strip/article/details/34066019

下面给出代码：

```
#include <bits/stdc++.h>
#define MAX 100010

using namespace std;

char Input()
{
    char str;
    scanf("%c",&str);
    while(str == ' ' || str == '\n')
        scanf("%c",&str);
    return str;
}

int work(int v,int *leaf)
{
    int temp, value;
    scanf("%d",&value);
    temp = Input();
    int max_num=0,left=0,right=0;
    if(temp == '(')
    {
        if(work(v-value,&left))  max_num=1;
        temp = Input();
        if(work(v-value,&right)) max_num=1;
        temp = Input();
        if(left&&right) max_num = (v == value);
    }
    else *leaf = 1;
    return max_num;
}
int main()
{
    int n,temp;
    while(~scanf("%d",&n))
    {
        Input();
        if(work(n,&temp))
            printf("yes\n");
        else
            printf("no\n");
    }
    return 0;
}
```

其他方法：

```
#include <iostream>
#include <string>
using namespace std;
//递归扫描输入的整棵树
bool ScanTree(int nSum, int nDest, bool *pNull) {
    static int nChild;
    //略去当前一级前导的左括号
    cin >> (char&)nChild;
    //br用于递归子节点的计算结果，bNull表示左右子是否为空
    bool br = false, bNull1 = false, bNull2 = false;
    //如果读入值失败，则该节点必为空
    if (!(*pNull = ((cin >> nChild) == 0))) {
        //总和加上读入的值，遍例子节点
        nSum += nChild;
```

```
        //判断两个子节点是否能返回正确的结果
        br = ScanTree(nSum, nDest, &bNull1) | ScanTree(nSum, nDest, &bNull2);
        //如果两个子节点都为空，则本节点为叶，检验是否达到目标值
        if (bNull1 && bNull2) {
            br = (nSum == nDest);
        }
    }
    //清除节点为空时cin的错误状态
    cin.clear();
    //略去当前一级末尾的右括号
    cin >> (char&)nChild;
    return br;
}
//主函数
int main(void) {
    bool bNull;
    //输入目标值
    for (int nDest; cin >> nDest;) {
        //根据结果输出yes或no
        cout << (ScanTree(0, nDest, &bNull) ? "yes" : "no") << endl;
    }
    return 0;
}
```

使用栈解决的代码：

```
#include <stdio.h>
#include <string.h>
#define MAXN 10000

int stack[MAXN];
int topc, top, t;

bool judge() {
    int sum = 0;
    for (int i=1; i<=top; i++)
        sum += stack[i];
    if (sum == t)
        return true;
    return false;
}

int main() {

    //freopen("f:\\out.txt", "w", stdout);
    while (scanf("%d", &t) != EOF) {
        int tmp = 0, flag = 0, isNeg = 0;
        char pre[4];
        topc = top = 0;
        memset(pre, 0, sizeof (pre));

        while (1) {
            // 接收字符的代码，忽略掉空格和换行
            char ch = getchar();
            while ('\n'==ch || ' '==ch)
                ch = getchar();

            // 记录该字符前三个字符，便于判断是否为叶子
            pre[3] = pre[2];
            pre[2] = pre[1];
            pre[1] = pre[0];
            pre[0] = ch;

            // 如果遇到左括弧就进栈
            if ('(' == ch) {
                topc++;
                if (tmp) {
                    if (isNeg) {
                        tmp *= -1;
                        isNeg = 0;
                    }
                    stack[++top] = tmp;
                    tmp = 0;
```

```
48                  }
49                  continue;
50              }
51
52              // 如果遇到右括弧就出栈
53              if (')' == ch) {
54                  // 如果为叶子便计算
55                  if ('('==pre[1] && ')'==pre[2] && '('==pre[3]) {
56                      if (!flag)
57                          flag = judge();
58                  }
59                  else if (pre[1] != '('){
60                      top--;
61                  }
62                  topc--;
63                  // 如果左括弧都被匹配完说明二叉树输入完毕
64                  if (!topc)
65                      break;
66                  continue;
67              }
68              if ('-' == ch)
69                  isNeg = 1;
70              else
71                  tmp = tmp*10 + (ch-'0');
72          }
73
74          if (flag)
75              printf("yes\n");
76          else
77              printf("no\n");
78      }
79
80      return 0;
81  }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5049686>

LeetCode 2 Add Two Numbers（链表操作）

2016年5月6日 13:33

题目来源：https://leetcode.com/problems/add-two-numbers/

You are given two linked lists representing two non-negative numbers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)

Output: 7 -> 0 -> 8

题目要求：给出两个链表l1和l2,将相对应的元素相加,如果产生进位则保留个位,相应的十位加在后面的对应元素上.

解体思路：

首先判断链表是否为空,若l1空,直接返回l2,相反返回l2

然后找到l1和l2链表最短的一个长度min_len,进入循环

循环过程中,为了减少空间的开销,可以将链表l1作为最终的结果链表,只需要改变l1的最大容量为l1+l2即可,另外相加过程中需要一个整型temp变量来保存进位.

循环结束之后,判断l1或l2空否(之前计算最短链表时可以做标记),然后将不空的链表中其他元素全部放入结果链表中即可.

提交代码：

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */

class Solution {
public:
    ListNode *addTwoNumbers(ListNode *l1, ListNode *l2) {
        // Start typing your C/C++ solution below
        // DO NOT write int main() function
//        ListNode *pResult = NULL;
//        ListNode **pCur = &pResult;

        ListNode rootNode(0);
        ListNode *pCurNode = &rootNode;
        int a = 0;
        while (l1 || l2)
        {
            int v1 = (l1 ? l1->val : 0);
            int v2 = (l2 ? l2->val : 0);
            int temp = v1 + v2 + a;
            a = temp / 10;
            ListNode *pNode = new ListNode((temp % 10));
            pCurNode->next = pNode;
            pCurNode = pNode;
            if (l1)
                l1 = l1->next;
            if (l2)
                l2 = l2->next;
```

```
        }
        if (a > 0)
        {
            ListNode *pNode = new ListNode(a);
            pCurNode->next = pNode;
        }
        return rootNode.next;
    }
};
```

下面给出数组解决的代码,具体过程相同,只是数据存储结构不同:

```
#include <bits/stdc++.h>
#define MAX 1000010

using namespace std;

int main()
{
    int n1,n2;
    while(~scanf("%d %d",&n1,&n2))
    {
        int *a=new int[n1+n2+1];
        int *b=new int[n2];
        for(int i=0;i<n1;i++)
            scanf("%d",&a[i]);
        for(int i=0;i<n2;i++)
            scanf("%d",&b[i]);
        int minlen=min(n1,n2);
        int maxlen=n1+n2-minlen;
        int temp=0;
        for(int i=0;i<minlen;i++)
        {
            a[i]=a[i]+b[i]+temp;
            if(a[i]>=10)
            {
                temp=a[i]/10;
                a[i]=a[i]%10;
            }
        }
        for(int j=minlen;j<maxlen;j++)
        {
                if(maxlen==n1)
                    a[j]=a[j];
                else
                    a[j]=b[j];
        }
        for(int i=0;i<maxlen;i++)
            printf("%d ",a[i]);
        printf("\n");
    }
}
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5049623>

LeetCode 1 Two Sum（二分法）

2016年5月6日　　13:33

题目来源：https://leetcode.com/problems/two-sum/

Given an array of integers, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have exactly one solution.

Input: numbers={2, 7, 11, 15}, target=9

Output: index1=1, index2=2

解题思路：

题目要求：给出一个数组numbers 以及 目标和target. 要求找到数组中两个数相加等于target所对应的下标.(下标不为0！)

/*第一次做LeetCode不熟悉.一直写着int main(),一直给WA.（明明本地的测试已经过了）.之后才明白代码的要求.*/

具体方法：数组进行升序或降序排序（下面采用升序排序）,采用二分法进行寻找.

分为三种情况：

```
1 if(num[left].x+num[right].x==target)
2 else if(num[left].x+num[right].x>target)
3 else if(num[left].x+num[right].x<target)
```

相对应的操作：

```
if(num[left].x+num[right].x==target)
{
    l=num[num[left].sort_id].id;
    r=num[num[right].sort_id].id;
    break;
}//下面的left和right的移动取决于排序是按照升序还是降序
else if(num[left].x+num[right].x>target)
{
    right=right-1;
}
else if(num[left].x+num[right].x<target)
{
    left=left+1;
}
```

给出代码：

```
#include <bits/stdc++.h>
#define MAX 10010
using namespace std;
struct Node{
    int x;
    int id;
    int sort_id;
```

```
};
bool cmp(Node a,Node b)
{
    return a.x<b.x;
}
Node num[MAX];
int main()
{
    int n,target,l,r;
    while(~scanf("%d",&n))
    {
        for(int i=0;i<n;i++)
        {
            scanf("%d",&num[i].x);
            num[i].id=i+1;
        }
        scanf("%d",&target);
        sort(num,num+n,cmp);
        for(int i=0;i<n;i++)
        {
            num[i].sort_id=i;
        }
        int left=0,right=n-1;
        while(left<right)
        {
            if(num[left].x+num[right].x==target)
            {
                l=num[num[left].sort_id].id;
                r=num[num[right].sort_id].id;
                break;
            }
            else if(num[left].x+num[right].x>target)
            {
                right=right-1;
            }
            else
            {
                left=left+1;
            }
        }
        printf("%d %d\n",l,r);
    }
}
```

提交代码：

```
 1 class Solution {
 2 public:
 3     vector<int> twoSum(vector<int>& nums, int target) {
 4         vector<int> result;
 5         int n = nums.size();
 6         if(n < 2)
 7             return result;
 8         vector<int> original = nums;
 9         sort(nums.begin(), nums.end());
10         int left=0, right=n-1;
11         int i, j, smaller, bigger;
12         while(left < right)
13         {
14             if(nums[left]+nums[right] == target)
15             {
16                 for(i=0; i<n; i++)
17                 {
18                     if(nums[left] == original[i])
19                         {
20                             result.push_back(i+1);
21                             break;
22                         }
23                 }
24                 for(j=n-1; j>=0; j--)
25                 {
26                     if(nums[right] == original[j])
```

```
27                      {
28                          result.push_back(j+1);
29                          break;
30                      }
31                  }
32                  if(result[0] < result[1])
33                  {
34                      smaller = result[0];
35                      bigger  = result[1];
36                  }
37                  else
38                  {
39                      smaller = result[1];
40                      bigger  = result[0];
41                  }
42                  result[0] = smaller;
43                  result[1] = bigger;
44                  return result;
45              }
46              else if(nums[left]+nums[right] < target)
47                  left = left + 1;
48              else
49                  right = right - 1;
50          }
51          return result;
52      }
53  };
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5049571>

Uva 110 - Meta-Loopless Sorts（!循环，回溯！）

2016年5月6日 13:33

题目来源：https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=3&page=show_problem&problem=46

Meta-Loopless Sorts

Background

Sorting holds an important place in computer science. Analyzing and implementing various sorting algorithms forms an important part of the education of most computer scientists, and sorting accounts for a significant percentage of the world's computational resources. Sorting algorithms range from the bewilderingly popular Bubble sort, to Quicksort, to parallel sorting algorithms and sorting networks. In this problem you will be writing a program that creates a sorting program (a meta-sorter).

The Problem

The problem is to create several programs whose output is a standard Pascal program that sorts *n* numbers where *n* is the only input to the program you will write. The Pascal programs generated by your program must have the following properties:

- They must begin with program sort(input,output);
- They must declare storage for exactly *n* integer variables. The names of the variables must come from the first *n* letters of the alphabet (a,b,c,d,e,f).
- A single readln statement must read in values for all the integer variables.
- Other than writeln statements, the only statements in the program are if then else statements. The boolean conditional for each if statement must consist of one strict inequality (either < or >) of two integer variables.
  Exactly *n*! writeln statements must appear in the program.
- Exactly three semi-colons must appear in the programs

1. after the program header: program sort(input,output);
2. after the variable declaration: ...: integer;
3. after the readln statement: readln(...);

- No redundant comparisons of integer variables should be made. For example, during program execution, once it is determined that $a < b$, variables *a* and *b* should not be compared again.
- Every writeln statement must appear on a line by itself.
- The programs must compile. Executing the program with input consisting of any arrangement of any *n* distinct integer values should result in the input values being printed in sorted order.

For those unfamiliar with Pascal syntax, the example at the end of this problem completely defines the small subset of Pascal needed.

The Input

The input consist on a number in the first line indicating the number M of programs to make, followed by a blank line. Then there are M test cases, each one consisting on a single integer *n* on a line by itself with 1

$\leq$

n

$\leq$

8. There will be a blank line between test cases.

The Output

The output is M compilable standard Pascal programs meeting the criteria specified above. Print a blank line between two consecutive programs.

Sample Input

```
1
3
```

Sample Output

```
program sort(input,output);
var
a,b,c : integer;
begin
  readln(a,b,c);
  if a < b then
    if b < c then
      writeln(a,b,c)
    else if a < c then
      writeln(a,c,b)
    else
      writeln(c,a,b)
  else
    if a < c then
      writeln(b,a,c)
    else if b < c then
      writeln(b,c,a)
    else
      writeln(c,b,a)
end.
```

Miguel Revilla 2001-05-25 推荐博客：
http://www.cnblogs.com/java20130723/p/3212108.html 代码：（没有缩进处理）

```
#include <cstdio>
const int maxn = 10;

int n, arr[maxn];

void insert_sort(int p, int c) {         //插入排序
    for (int i = c; i > p; i--)
        arr[i] = arr[i - 1];
    arr[p] = c;
}

int dfs(int d) {
    int tmp[d + 1];                          //创建数组储存原来的数值，不然会乱掉
    for (int j = 1; j <= n; j++)
        tmp[j] = arr[j];
    for (int i = d; i >= 1; i--) {             //循环从现排好的串后序进行dfs
        printf("if %c < %c then\n", arr[i] + 'a' - 1, d + 'a');
        insert_sort(i + 1, d + 1);             //将接下去的字母插入到i+1的位置
        if (d + 1 == n) {                      //dfs到最深处，输出
            printf("writeln(");
            printf("%c", arr[1] + 'a' - 1);
            for (int j = 2; j <= d + 1; j++)
                printf(",%c", arr[j] + 'a' - 1);
            printf(")\n");
            printf("else\n");
        }
        else {
            dfs(d + 1);
            printf("else\n");
        }
        for (int j = 1; j <= n; j++)         //还原数组
            arr[j] = tmp[j];
    }
    insert_sort(1, d + 1);                     //下面是对最后一个情况，即字母插到整个数组前面，这里是没有else的
    if (d + 1 == n) {
        printf("writeln(");
        printf("%c", arr[1] + 'a' - 1);
        for (int j = 2; j <= d + 1; j++)
            printf(",%c", arr[j] + 'a' - 1);
        printf(")\n");
    }
    else
        dfs(d + 1);
    for (int i = 1; i <= n; i++)
        arr[i] = tmp[i];
}

int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n);
        //前面部分
        printf("program sort(input,output);\nvar\n");
        printf("a");
        for (int i = 2; i <= n; i++)
            printf(",%c", i + 'a' - 1);
        printf(" : integer;\nbegin\nreadln(");
        printf("a");
        for (int i = 2; i <= n; i++)
            printf(",%c", i + 'a' - 1);
        printf(");\n");
        dfs(0);                  //开始深搜
        printf("end.\n");
        if (t != 0)
```

```
66              printf("\n");
67      }
68      return 0;
69  }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5049541>

UVa 109 - SCUD Busters（凸包计算）

2016年5月6日 13:34

题目来源：https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=3&page=show_problem&problem=45

SCUD Busters

Background

Some problems are difficult to solve but have a simplification that is easy to solve. Rather than deal with the difficulties of constructing a model of the Earth (a somewhat oblate spheroid), consider a pre-Columbian flat world that is a 500 kilometer

500 kilometer square.

In the model used in this problem, the flat world consists of several warring kingdoms. Though warlike, the people of the world are strict isolationists; each kingdom is surrounded by a high (but thin) wall designed to both protect the kingdom and to isolate it. To avoid fights for power, each kingdom has its own electric power plant.

When the urge to fight becomes too great, the people of a kingdom often launch missiles at other kingdoms. Each SCUD missile (Sanitary Cleansing Universal Destroyer) that lands within the walls of a kingdom destroys that kingdom's power plant (without loss of life).

The Problem

Given coordinate locations of several kingdoms (by specifying the locations of

houses and the location of the power plant in a kingdom) and missile landings you are to write a program that determines the total area of all kingdoms that are without power after an exchange of missile fire.

In the simple world of this problem kingdoms do not overlap. Furthermore, the walls surrounding each kingdom are considered to be of zero thickness. The wall surrounding a kingdom is the minimal-perimeter wall that completely surrounds all the houses and the power station that comprise a kingdom; the area of a kingdom is the area enclosed by the minimal-perimeter thin wall.

There is exactly one power station per kingdom.

There may be empty space between kingdoms.

The Input

The input is a sequence of kingdom specifications followed by a sequence of missile landing locations.

A kingdom is specified by a number *N* ($3 \leq N \leq 100$) on a single line which indicates the number of sites in this kingdom. The next line contains the *x* and *y* coordinates of the power station, followed by *N*-1 lines of *x*, *y* pairs indicating the locations of homes served by this power station. A value of -1 for *N* indicates that there are no more kingdoms. There will be at least one kingdom in the data set.

Following the last kingdom specification will be the coordinates of one or more missile attacks, indicating the location of a missile landing. Each missile location is on a line by itself. You are to process missile attacks until you reach the end of the file.

Locations are specified in kilometers using coordinates on a 500 km by 500 km grid. All coordinates will be integers between 0 and 500 inclusive. Coordinates are specified as a pair of integers separated by white-space on a single line. The input file will consist of up to 20 kingdoms, followed by any number of missile attacks.

The Output

The output consists of a single number representing the total area of all kingdoms without electricity after all missile attacks have been processed. The number should be printed with (and correct to) two decimal places.

Sample Input

```
12
3 3
4 6
4 11
4 8
10 6
5 7
6 6
6 3
7 9
10 4
10 9
1 7
5
20 20
20 40
40 20
40 40
30 30
3
10 10
21 10
21 13
-1
5 5
20 12
```

Sample Output

```
70.50
```

A Hint

You may or may not find the following formula useful.

Given a polygon described by the vertices
$v_0, v_1, \ldots, v_n$

such that
$v_0 = v_n$

, the signed area of the polygon is given by

$$a = \frac{1}{2} \sum_{i=1}^{n} (x_{i-1} y_i) - (x_i y_{i-1})$$

where the x, y coordinates of
$v_i = (x_i, y_i)$

; the edges of the polygon are from
v_i

to
v_{i+1}

for
$i = 0 \ldots n-1$

.

If the points describing the polygon are given in a counterclockwise direction, the value of *a* will be positive, and if the points of the polygon are listed in a clockwise direction, the value of *a* will be negative.

推荐博客：
http://www.cnblogs.com/devymex/archive/2010/08/09/1795391.html

解题思路：

计算几何类型的题目。需要用到三个基本算法，一是求凸包，二是判断点在多边形内，三是求多边形面积（题目中已给出）。关于凸包算法请详见Graham's Scan法。判断点在多边形内的算法有很多种，这里用到的是外积法：设待判断的点为p，逆时针或顺时针遍例多边形的每个点v_n，将两个

向量<p, v_n>和<v_n, v_{n+1}>做外积。如果对于多边形上所有的点，外积的符号都相同（顺时针为负，逆时针为正），则可断定p在多边形内。外积出现0，则表示p在边上，否则在多边形外。

算法的思路很直接，实现也很简单，关键是这道题的测试数据太扯蛋了，让我郁闷了很久。题目中并未说明导弹打在墙上怎么办，只是说“... whithin the wall ...”。根据测试结果来看，打在墙上和打在据点上都要算打中。题目中还提到国家不会互相重叠“... kingdoms do not overlap.”，但测试表明数据里确有重叠的情况，因此在导弹击中后一定要跳出循环，否则会出现一弹多击的情况。

给你N个王国，求下凸包，再求面积。给你一些炮弹，问炮弹炸掉的面积。（一个炮弹炸的话，整个王国都被炸了）。

直接求凸包后，求出各个王国的面积，然后判断炮弹在哪个王国里，这个直接用判断点是否在多边形内。

参考代码：

```
#include<bits/stdc++.h>

using namespace std;

const int MAX = 110;
struct point{ double x,y;};         //点
struct polygon{ point c[MAX],a; double area; int n;};
struct segment{ point a,b;};         // 线段
const double eps = 1e-6;
bool dy(double x,double y)     {     return x > y + eps;}     // x > y
bool xy(double x,double y)     {     return x < y - eps;}     // x < y
bool dyd(double x,double y)     {      return x > y - eps;}     // x >= y
bool xyd(double x,double y)     {     return x < y + eps;}      // x <= y
bool dd(double x,double y)      {     return fabs( x - y ) < eps;}  // x == y
polygon king[MAX];
point c[MAX];
double crossProduct(point a,point b,point c)//向量 ac 在 ab 的方向
{
    return (c.x - a.x)*(b.y - a.y) - (b.x - a.x)*(c.y - a.y);
}
double disp2p(point a,point b)
{
    return sqrt( ( a.x - b.x ) * ( a.x - b.x ) + ( a.y - b.y ) * ( a.y -
b.y ) );
}
double area_polygon(point p[],int n)
{
    double s = 0.0;
    for(int i=0; i<n; i++)
        s += p[(i+1)%n].y * p[i].x - p[(i+1)%n].x * p[i].y;
    return fabs(s)/2.0;
}
bool cmp(point a,point b)  // 排序
{
    double len = crossProduct(c[0],a,b);
    if( dd(len,0.0) )
        return xy(disp2p(c[0],a),disp2p(c[0],b));
    return xy(len,0.0);
}
bool onSegment(point a, point b, point c)
{
    double maxx = max(a.x,b.x);
    double maxy = max(a.y,b.y);
```

```
 43     double minx = min(a.x,b.x);
 44     double miny = min(a.y,b.y);
 45     if( dd(crossProduct(a,b,c),0.0) && dyd(c.x,minx) && xyd(c.x,maxx) &&
dyd(c.y,miny) && xyd(c.y,maxy) )
 46         return true;
 47     return false;
 48 }
 49 bool segIntersect(point p1,point p2, point p3, point p4)
 50 {
 51     double d1 = crossProduct(p3,p4,p1);
 52     double d2 = crossProduct(p3,p4,p2);
 53     double d3 = crossProduct(p1,p2,p3);
 54     double d4 = crossProduct(p1,p2,p4);
 55     if( xy(d1 * d2,0.0) && xy(d3 * d4,0.0) )    return true;
 56     if( dd(d1,0.0) && onSegment(p3,p4,p1) )        return true;//如果不判端点相
交，则下面这四句话不需要
 57     if( dd(d2,0.0) && onSegment(p3,p4,p2) )        return true;
 58     if( dd(d3,0.0) && onSegment(p1,p2,p3) )        return true;
 59     if( dd(d4,0.0) && onSegment(p1,p2,p4) )        return true;
 60     return false;
 61 }
 62 bool point_inPolygon(point pot,point p[],int n)
 63 {
 64     int count = 0;
 65     segment l;
 66     l.a = pot;
 67     l.b.x = 1e10*1.0;
 68     l.b.y = pot.y;
 69     p[n] = p[0];
 70     for(int i=0; i<n; i++)
 71     {
 72         if( onSegment(p[i],p[i+1],pot) )
 73             return true;
 74         if( !dd(p[i].y,p[i+1].y) )
 75         {
 76             int tmp = -1;
 77             if( onSegment(l.a,l.b,p[i]) )
 78                 tmp = i;
 79             else
 80                 if( onSegment(l.a,l.b,p[i+1]) )
 81                     tmp = i+1;
 82             if( tmp != -1 && dd(p[tmp].y,max(p[i].y,p[i+1].y)) )
 83                 count++;
 84             else
 85                 if( tmp == -1 && segIntersect(p[i],p[i+1],l.a,l.b) )
 86                     count++;
 87         }
 88     }
 89     if( count % 2 == 1 )
 90         return true;
 91     return false;
 92 }
 93 point stk[MAX];
 94 int top;
 95 double Graham(int n)
 96 {
 97     int tmp = 0;
 98     for(int i=1; i<n; i++)
 99         if( xy(c[i].x,c[tmp].x) || dd(c[i].x,c[tmp].x) && xy(c[i].y,c[tmp].y) )
100             tmp = i;
101     swap(c[0],c[tmp]);
102     sort(c+1,c+n,cmp);
103     stk[0] = c[0]; stk[1] = c[1];
104     top = 1;
105     for(int i=2; i<n; i++)
106     {
107         while( xyd( crossProduct(stk[top],stk[top-1],c[i]), 0.0 ) && top >= 1 )
108             top--;
109         stk[++top] = c[i];
110     }
111     return area_polygon(stk,top+1);
112 }
113 int main()
114 {
115     int n;
```

```
116     int i = 0;
117     double x,y;
118     while( ~scanf("%d",&n) && n != -1 )
119     {
120         king[i].n = n;
121         for(int k=0; k<n; k++)
122             scanf("%lf %lf",&king[i].c[k].x,&king[i].c[k].y);
123         king[i].a = king[i].c[0];
124         i++;
125     }
126
127     double sum = 0.0;
128     for(int k=0; k<i; k++)
129     {
130         memcpy(c,king[k].c,sizeof(king[k].c));
131         king[k].area = Graham(king[k].n);
132         memcpy(king[k].c,stk,sizeof(stk));
133         king[k].n = top+1;
134     }
135     point pot;
136     bool die[MAX];
137     memset(die,false,sizeof(die));
138     while( ~scanf("%lf %lf",&pot.x,&pot.y) )
139     {
140         for(int k=0; k<i; k++)
141             if( point_inPolygon(pot,king[k].c,king[k].n) && !die[k] )
142             {
143                 die[k] = true;
144                 sum += king[k].area;
145             }
146     }
147
148     printf("%.2lf\n",sum);
149 return 0;
150 }
```

参考代码2：

```
 1 #include <algorithm>
 2 #include <functional>
 3 #include <iomanip>
 4 #include <iostream>
 5 #include <vector>
 6 #include <math.h>
 7
 8 using namespace std;
 9
10 struct POINT {
11     int x; int y;
12     bool operator==(const POINT &other) {
13         return (x == other.x && y == other.y);
14     }
15 } ptBase;
16
17 typedef vector<POINT> PTARRAY;
18
19 bool CompareAngle(POINT pt1, POINT pt2) {
20     pt1.x -= ptBase.x, pt1.y -= ptBase.y;
21     pt2.x -= ptBase.x, pt2.y -= ptBase.y;
22     return (pt1.x / sqrt((float)(pt1.x * pt1.x + pt1.y * pt1.y)) <
23         pt2.x / sqrt((float)(pt2.x * pt2.x + pt2.y * pt2.y)));
24 }
25
26 void CalcConvexHull(PTARRAY &vecSrc, PTARRAY &vecCH) {
27     ptBase = vecSrc.back();
28     sort(vecSrc.begin(), vecSrc.end() - 1, &CompareAngle);
29     vecCH.push_back(ptBase);
30     vecCH.push_back(vecSrc.front());
31     POINT ptLastVec = { vecCH.back().x - ptBase.x,
32         vecCH.back().y - ptBase.y };
33     PTARRAY::iterator i = vecSrc.begin();
34     for (++i; i != vecSrc.end() - 1; ++i) {
35         POINT ptCurVec = { i->x - vecCH.back().x, i->y - vecCH.back().y };
36         while (ptCurVec.x * ptLastVec.y - ptCurVec.y * ptLastVec.x < 0) {
37             vecCH.pop_back();
```

```
38              ptCurVec.x = i->x - vecCH.back().x;
39              ptCurVec.y = i->y - vecCH.back().y;
40              ptLastVec.x = vecCH.back().x - (vecCH.end() - 2)->x;
41              ptLastVec.y = vecCH.back().y - (vecCH.end() - 2)->y;
42          }
43          vecCH.push_back(*i);
44          ptLastVec = ptCurVec;
45      }
46      vecCH.push_back(vecCH.front());
47  }
48
49  int CalcArea(PTARRAY &vecCH) {
50      int nArea = 0;
51      for (PTARRAY::iterator i = vecCH.begin(); i != vecCH.end() - 1; ++i) {
52          nArea += (i + 1)->x * i->y - i->x * (i + 1)->y;
53      }
54      return nArea;
55  }
56
57  bool PointInConvexHull(POINT pt, PTARRAY &vecCH) {
58      for (PTARRAY::iterator i = vecCH.begin(); i != vecCH.end() - 1; ++i) {
59          int nX1 = pt.x - i->x, nY1 = pt.y - i->y;
60          int nX2 = (i + 1)->x - i->x, nY2 = (i + 1)->y - i->y;
61          if (nX1 * nY2 - nY1 * nX2 < 0) {
62              return false;
63          }
64      }
65      return true;
66  }
67
68  int main(void) {
69      vector<PTARRAY> vecKingdom;
70      POINT ptIn;
71      int aFlag[100] = {0}, nAreaSum = 0;
72      for (int nPtCnt; cin >> nPtCnt && nPtCnt >= 1;) {
73          PTARRAY vecSrc, vecCH;
74          cin >> ptIn.x >> ptIn.y;
75          vecSrc.push_back(ptIn);
76          for (; --nPtCnt != 0;) {
77              cin >> ptIn.x >> ptIn.y;
78              POINT &ptMin = vecSrc.back();
79              vecSrc.insert(vecSrc.end() - (ptIn.y > ptMin.y ||
80                  (ptIn.y == ptMin.y && ptIn.x > ptMin.x)), ptIn);
81          }
82          CalcConvexHull(vecSrc, vecCH);
83          vecKingdom.push_back(vecCH);
84      }
85      while (cin >> ptIn.x >> ptIn.y) {
86          vector<PTARRAY>::iterator i = vecKingdom.begin();
87          for (int k = 0; i != vecKingdom.end(); ++i, ++k) {
88              if (PointInConvexHull(ptIn, *i) && aFlag[k] != 1) {
89                  nAreaSum += CalcArea(*i);
90                  aFlag[k] = 1;
91                  break;
92              }
93          }
94      }
95      cout << setiosflags(ios::fixed) << setprecision(2);
96      cout << (float)nAreaSum / 2.0f << endl;
97      return 0;
98  }
```

来自 <http://i.cnblogs.com/EditPosts.aspx?postid=5049517>