راهنمایی الگوریتم :

تعریف ها :

- **s نقطه شروع بازه مورد نظر است و e نقطه پایان بازه مورد نظر است**
- **length طول بازه است**

تعریف سگمنت در نقطه [x,x] : از نقطه x تا نقطه **length** (یعنی نقطه پایان) تمام نقاط توسط بازه هایی که ما انتخاب کرده ایم دیده شده اند و تعداد روش هایی که میتوانیم این کار را انجام دهیم در سگمنت x,x می نویسیم

مقدار سگمنت در یک نقطه برابر با تعداد روش هایی است که میتوانیم از نقطه x تا **length** را پوشش دهیم

- سگمنت [l,r] یعنی مجموع مقدار های سگمنت های [x,x] به طوری که x از l بزرگتر مساوی باشد و x از r کوچکتر مساوی باشد

لیزی : برای اینکه مقدار سگمنت [l,r] ، به اصطلاح up-to-date باشد باشد مقدار سگمنت [l,r] را باید در $2^{\text{lazy}[l,r]}$ ضرب بکنیم و بعد با توجه به ویژگی های عمومی لیزی عملیات را برای دو بچه لیزی انجام بدهیم و بعد از آن lazy[l,r] را برابر با صفر قرار بدهیم

نحوه اجرای الگوریتم :

ابتدا بر اساس پایان بازه ها عملیات مرتب سازی از زیاد به کم را بر روی بازه ها انجام میدهیم (استفاده از توابع آماده sort برای این کار کافی است).

مقدار تمام خانه های seg و lazy در ابتدا کار برابر با صفر است

سپس مقدار seg[**length**, **length**] را برابر با یک قرار میدهیم چون یک بازه تهی وجود دارد که میتواند از نقطه **length** تا نقطه **length** را ببیند و طبق تعریف است. (بدیهی است که باید این مقداردهی توسط تابع update انجام شود در غیر اینصورت مقدار seg[0, **length**] که برابر با مجموع سگمنت های [x,x] است که x بین ۰ تا **length** است صفر خواهد ماند که اشتباه است)

عملیات زیر را به ازای بازه x که x از ۰ تا n انجام میدهیم (منظور از این شماره ها بازه های سورت شده است و بازه صفرم یعنی بازه ای که بیشترین E را دارد)

منظور یک حلقه روی بازه ها است

دو حالت برای بازه x وجود دارد

1. بازه x انتخاب نمیشود :
در این حالت با انتخاب نکردن این بازه تمام seg ها همان مقدار قبلی باقی میمانند پس موقعی که به این حالت میرسیم سگمنت های فعلی و جواب های موجود جواب همین حالت است پس هیچ کاری نیاز نیست انجام بدهیم
2. بازه x انتخاب میشود :
با توجه به نقاط segment سه حالت امکانپذیر است اگر j را به عنوان مقدار های segment قبل از این بازه x در نظر بگیریم داریم :
$S \le j \le e$ در این حالت seg[s,e] روش وجود دارد که در تمام این روش ها ما بازه هایی را انتخاب کرده ایم که آنها از نقطه j تا نقطه **length** را دیده اند که j با توجه به شماره مورد نظر میتواند بین s و e باشد
پس با انتخاب بازه x و برداشتن آن ما seg[s,s] را خواهیم دید . خط زیر را میدانیم:

$$seg[s,e] = seg[s,s] + seg[s+1,s+1] + seg[s+2,s+2] + \cdots + seg[e,e]$$

$$seg[s,s] \mathrel{+}= seg[s,e]$$

$0 \le j \le (s-1)$ با آمدن این بازه تاثیری بر سگمنت های آن ندارد و میتوان براحتی بررسی کرد که در این حالت ما فقط کافی است مقدار های seg[j,j] را ضربدر دو بکنیم

$e < j$ ما نیاز نداریم که کاری بکنیم (اثبات برای علاقه مندان)

اثبات : چون در این حالت ما از **length** تا z را اوکی کرده بودیم و با برداشتن این بازه از s تا eرا اوکی میکنیم و بازه z تا e توسط هیچ بازه ای اوکی نشده است و در آینده هم نخواهد شد چون ما سرت مان را بر اساس e ها کرده بودیم و e های آینده از e این بازه قطعا کمتر هستند

در نهایت طبق تعریف و اینکه ما سگمنت ها را به درستی محاسبه کرده ایم باید مقدار seg[0,0] را به عنوان خروجی بدهیم که یعنی ما بازه ۰ تا **length** را به درستی توسط بازه ها پوشش داده ایم و مقدار نهایی را محاسبه کرده ایم و به عنوان خروجی سوال آن را پرینت میکنیم.

راهنمایی پیاده سازی :

در نهایت با توجه به تعریف سگمنت و استفاده از توابع query و update و update2 میتوانیم به جواب موردنظرمان برسیم

من با دو تابع برای update این کار را انجام دادم اولی برای جمع کردن قرار دادم و دومی را برای ضرب کردن قرار دادم توجه کنید که اگر با تابع lazy این کار را انجام ندهید چون در مرحله update کردن باید تمام زیر بازه ها را به درستی update بکنید پس از لحاظ زمانی به مشکل برخواهید شد چون این عملیات در O(n) هست و با lazy میتوان در O(log(n)) انجام داد و هدف از سگمنت تری زدن این سوال برای محاسبه کل در زمان O(nlog(n)) است وگرنه محاسبه در زمان O(n^2) بسیار راحت تر از این روش میباشد .

موفق باشید

Creating model managers 25
Building list and detail views 26
Creating list and detail views 26
Adding URL patterns for your views 28
Canonical URLs for models 29
Creating templates for your views 30
Adding pagination 34
Using class-based views 36
Summary 38
Chapter 2: Enhancing Your Blog with Advanced Features 39
Sharing posts by email 40
Creating forms with Django 40
Handling forms in views 41
Sending emails with Django 43
Rendering forms in templates 45
Creating a comment system 50
Building a model 50
Creating forms from models 52
Handling ModelForms in views 53
Adding comments to the post detail template 54
Adding the tagging functionality 58
Retrieving posts by similarity 64
Summary 66
Chapter 3: Extending Your Blog Application 67
Creating custom template tags and filters 68
Custom template tags 68
Custom template filters 73
Adding a sitemap to your site 76
Creating feeds for your blog posts 80
Adding full-text search to your blog 82
Installing PostgreSQL 83
Simple search lookups 84
Searching against multiple fields 84
Building a search view 85
Stemming and ranking results 88
Weighting queries 89
Searching with trigram similarity 90
Other full-text search engines 91
Summary 91

Chapter 4: Building a Social Website 93
Creating a social website project 94
Starting your social website project 94
Using the Django authentication framework 95
Creating a login view 96
Using Django authentication views 101
Login and logout views 102
Changing password views 108
Resetting password views 110
User registration and user profiles 115
User registration 115
Extending the user model 119
Using a custom user model 125
Using the messages framework 125
Building a custom authentication backend 128
Adding social authentication to your site 130
Running the development server through HTTPS 132
Authentication using Facebook 134
Authentication using Twitter 140
Authentication using Google 142
Summary 147
Chapter 5: Sharing Content on Your Website 149
Creating an image bookmarking website 150
Building the image model 150
Creating many-to-many relationships 152
Registering the image model in the administration site 153
Posting content from other websites 153
Cleaning form fields 154
Overriding the save() method of a ModelForm 155
Building a bookmarklet with jQuery 160
Creating a detail view for images 168
Creating image thumbnails using easy-thumbnails 170
Adding AJAX actions with jQuery 172
Loading jQuery 173
Cross-site request forgery in AJAX requests 174
Performing AJAX requests with jQuery 176
Creating custom decorators for your views 179
Adding AJAX pagination to your list views 181
Summary 186

Chapter 6: Tracking User Actions 187
Building a follow system 187
Creating many-to-many relationships with an intermediary model 188
Creating list and detail views for user profiles 191
Building an AJAX view to follow users 196
Building a generic activity stream application 198
Using the contenttypes framework 200
Adding generic relations to your models 201
Avoiding duplicate actions in the activity stream 204
Adding user actions to the activity stream 205
Displaying the activity stream 206
Optimizing QuerySets that involve related objects 207
Using select_related() 207
Using prefetch_related() 208
Creating templates for actions 208
Using signals for denormalizing counts 210
Working with signals 211
Application configuration classes 213
Using Redis for storing item views 215
Installing Redis 215
Using Redis with Python 217
Storing item views in Redis 218
Storing a ranking in Redis 220
Next steps with Redis 223
Summary 223
Chapter 7: Building an Online Shop 225
Creating an online shop project 226
Creating product catalog models 227
Registering catalog models on the administration site 229
Building catalog views 230
Creating catalog templates 233
Building a shopping cart 237
Using Django sessions 238
Session settings 239
Session expiration 240
Storing shopping carts in sessions 240
Creating shopping cart views 245
Adding items to the cart 245
Building a template to display the cart 247
Adding products to the cart 249
Updating product quantities in the cart 251

Creating a context processor for the current cart 252
Context processors 252
Setting the cart into the request context 253
Registering customer orders 255
Creating order models 255
Including order models in the administration site 257
Creating customer orders 258
Launching asynchronous tasks with Celery 263
Installing Celery 263
Installing RabbitMQ 264
Adding Celery to your project 264
Adding asynchronous tasks to your application 265
Monitoring Celery 267
Summary 268
Chapter 8: Managing Payments and Orders 269
Integrating a payment gateway 269
Creating a Braintree sandbox account 270
Installing the Braintree Python module 271
Integrating the payment gateway 272
Integrating Braintree using Hosted Fields 274
Testing payments 280
Going live 283
Exporting orders to CSV files 284
Adding custom actions to the administration site 284
Extending the administration site with custom views 287
Generating PDF invoices dynamically 292
Installing WeasyPrint 292
Creating a PDF template 292
Rendering PDF files 294
Sending PDF files by email 297
Summary 300
Chapter 9: Extending Your Shop 301
Creating a coupon system 301
Building the coupon model 302
Applying a coupon to the shopping cart 304
Applying coupons to orders 312
Adding internationalization and localization 314
Internationalization with Django 315
Internationalization and localization settings 315
Internationalization management commands 316
How to add translations to a Django project 316

How Django determines the current language 316
Preparing your project for internationalization 317
Translating Python code 318
Standard translations 319
Lazy translations 319
Translations including variables 319
Plural forms in translations 319
Translating your own code 320
Translating templates 324
The {% trans %} template tag 324
The {% blocktrans %} template tag 324
Translating the shop templates 325
Using the Rosetta translation interface 328
Fuzzy translations 331
URL patterns for internationalization 332
Adding a language prefix to URL patterns 332
Translating URL patterns 333
Allowing users to switch language 334
Translating models with django-parler 336
Installing django-parler 336
Translating model fields 337
Integrating translations into the administration site 339
Creating migrations for model translations 340
Adapting views for translations 341
Format localization 344
Using django-localflavor to validate form fields 345
Building a recommendation engine 347
Recommending products based on previous purchases 347
Summary 356
Chapter 10: Building an E-Learning Platform 357
Setting up the e-learning project 358
Building the course models 359
Registering the models in the administration site 361
Using fixtures to provide initial data for models 362
Creating models for diverse content 365
Using model inheritance 366
Abstract models 366
Multi-table model inheritance 367
Proxy models 367
Creating the content models 368
Creating custom model fields 370
Adding ordering to module and content objects 372
Creating a CMS 377
Adding an authentication system 377

Creating the authentication templates 378
Creating class-based views 381
Using mixins for class-based views 381
Working with groups and permissions 383
Restricting access to class-based views 385
Managing course modules and their contents 391
Using formsets for course modules 391
Adding content to course modules 396
Managing modules and their contents 402
Reordering modules and their contents 407
Using mixins from django-braces 407
Summary 411
Chapter 11: Rendering and Caching Content 413
Displaying courses 414
Adding student registration 419
Creating a student registration view 419
Enrolling on courses 422
Accessing the course contents 425
Rendering different types of content 429
Using the cache framework 432
Available cache backends 432
Installing Memcached 433
Cache settings 434
Adding Memcached to your project 434
Monitoring Memcached 435
Cache levels 436
Using the low-level cache API 436
Caching based on dynamic data 438
Caching template fragments 440
Caching views 441
Using the per-site cache 441
Summary 442
Chapter 12: Building an API 443
Building a RESTful API 444
Installing Django REST framework 444
Defining serializers 445
Understanding parsers and renderers 446
Building list and detail views 447
Creating nested serializers 450
Building custom API views 452
Handling authentication 453

Adding permissions to views 454
Creating viewsets and routers 456
Adding additional actions to viewsets 457
Creating custom permissions 458
Serializing course contents 459
Consuming the REST API 461
Summary 465
Chapter 13: Building a Chat Server 467
Creating a chat application 467
Implementing the chat room view 468
Deactivating the per-site cache 471
Real-time Django with Channels 471
Asynchronous applications using ASGI 471
The request/response cycle using Channels 472
Installing Channels 473
Writing a consumer 476
Routing 477
Implementing the WebSocket client 478
Enabling a channel layer 484
Channels and groups 484
Setting up a channel layer with Redis 484
Updating the consumer to broadcast messages 486
Adding context to the messages 490
Modifying the consumer to be fully asynchronous 494
Integrating the chat with existing views 495
Summary 496
Chapter 14: Going Live 497
Creating a production environment 497
Managing settings for multiple environments 498
Using PostgreSQL 500
Checking your project 501
Serving Django through WSGI 501
Installing uWSGI 502
Configuring uWSGI 502
Installing NGINX 505
The production environment 506
Configuring NGINX 506
Serving static and media assets 509
Securing connections with SSL/TLS 511
Creating an SSL/TLS certificate 511

Configuring NGINX to use SSL/TLS 512
Configuring your Django project for SSL/TLS 514
Redirecting HTTP traffic over to HTTPS 515
Using Daphne for Django Channels 516
Using secure connections for WebSockets 517
Including Daphne in the NGINX configuration 518
Creating a custom middleware 520
Creating a subdomain middleware 522
Serving multiple subdomains with NGINX 523
Implementing custom management commands 524
Summary 527
Other Books You May Enjoy 529
Index 533

Preface

Django is a powerful Python web framework that encourages rapid development and clean, pragmatic design, while offering a relatively shallow learning curve. This makes it attractive to both novice and expert programmers.

This book will guide you through the entire process of developing professional web applications with Django. The book not only covers the most relevant aspects of the framework, but it will also teach you how to integrate other popular technologies into your Django projects.

The book will walk you through the creation of real-world applications, solving common problems, and implementing best practices, using a step-by-step approach that is easy to follow.

After reading this book, you will have a good understanding of how Django works and how to build practical, advanced web applications.

Who this book is for

This book is intended for developers with Python knowledge who wish to learn Django in a pragmatic way. Perhaps you are completely new to Django, or you already know a little but you want to get the most out of it. This book will help you to master the most relevant areas of the framework by building practical projects from scratch. You need to have familiarity with programming concepts in order to read this book. Some previous knowledge of HTML and JavaScript is assumed.