《代码审计报告》

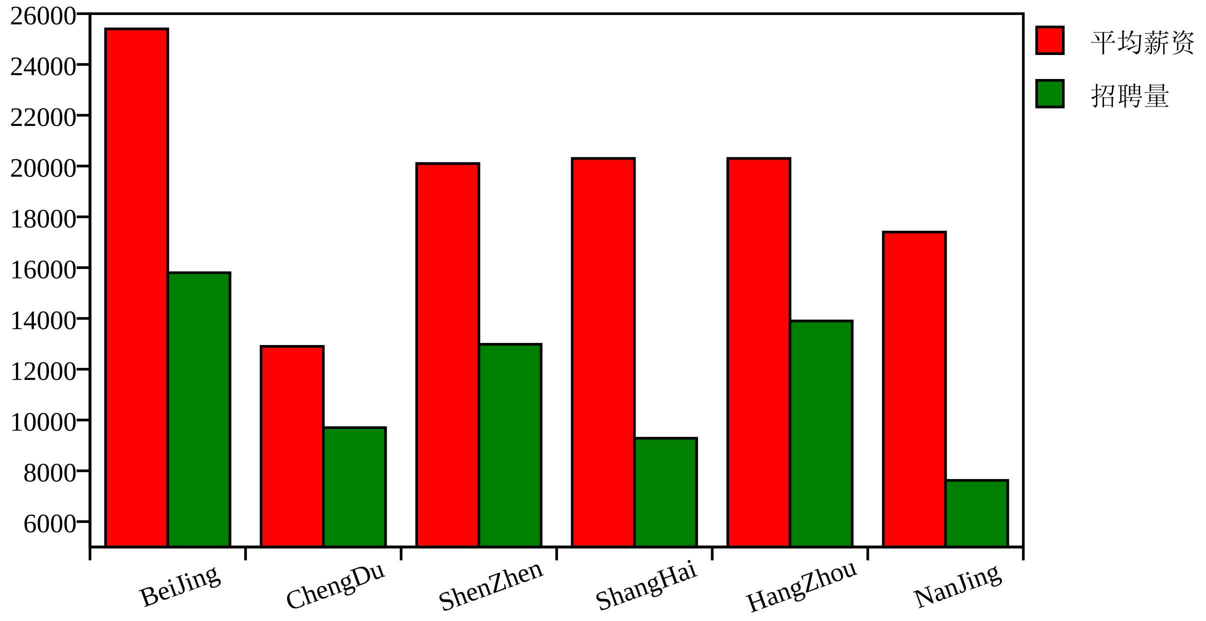
[风险函数统计结果]

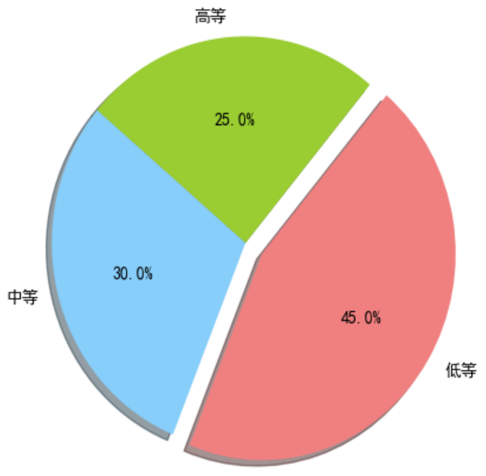
风险等级	风险数量
高等风险函数	2
中等风险函数	4
低等风险函数	10

[各风险函数详细信息]

风险等级	风险数量
高等风险函数	2
中等风险函数	4
低等风险函数	10

热门城市的就业情况





```
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:1:0: information: Include file: not found. Please note: Cppcheck does not need standard library headers to get proper results. [missingIncludeSystem]
#include
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:3:0: information: Include file: not found. Please note: Cppcheck does not need standard library headers to get proper results. [missingIncludeSystem]
#include
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:4:0: information: Include file: not found. Please note: Cppcheck does not need standard library headers to get proper results. [missingIncludeSystem]
#include
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:32:9: style: The scope of the variable 'begin' can be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an example where cppcheck will write that the scope for 'i' can be reduced:
void f(int x)
```

```
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0;' here
for (int n = 0; n < 10; ++n) {
// it is possible but not safe to move 'int i = 0;' here
do_something(&i;);
}
}
}
```

When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]

```
int i, begin, end,weight;
^
```

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:32:16: style: The scope of the variable 'end' can be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an example where cppcheck will write that the scope for 'i' can be reduced:

```
void f(int x)
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0;' here
for (int n = 0; n < 10; ++n) {
// it is possible but not safe to move 'int i = 0;' here
do_something(&i;);
}
}
}
When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]
int i, begin, end,weight;
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:32:20: style: The scope of the variable 'weight' can
be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an
example where cppcheck will write that the scope for 'i' can be reduced:
void f(int x)
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0;' here
for (int n = 0; n < 10; ++n) {
// it is possible but not safe to move 'int i = 0;' here
do_something(&i;);
}
}
}
When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]
int i, begin, end,weight;
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:59:6: style: The scope of the variable 'ver_data' can
be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an
example where cppcheck will write that the scope for 'i' can be reduced:
```

```
void f(int x)
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0;' here
for (int n = 0; n < 10; ++n) {
// it is possible but not safe to move 'int i = 0;' here
do_something(&i;);
}
}
}
When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]
int ver_data;
^
D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:104:28: style: The scope of the variable 'temp' can be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an example where cppcheck will write that the scope for 'i' can be reduced:
void f(int x)
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0;' here
for (int n = 0; n < 10; ++n) {
// it is possible but not safe to move 'int i = 0;' here
do_something(&i;);
}

}

}

When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]

int front = -1, rear = -1,temp;

^

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:126:6: style: The scope of the variable 'node' can be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an example where cppcheck will write that the scope for 'i' can be reduced:

void f(int x)

{

int i = 0;

if (x) {

// it's safe to move 'int i = 0;' here

for (int n = 0; n < 10; ++n) {

// it is possible but not safe to move 'int i = 0;' here

do_something(&i;);

}

}

}

When you see this message it is always safe to reduce the variable scope 1 level. [variableScope]

ver node;

^

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:164:17: style: Variable 'mincost' is assigned a value that is never used. [unreadVariable]
```

```
int i,j,mincost=INT_MAX,minIndex;

^

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:201:20: style: Variable 'mincost' is assigned a value

that is never used. [unreadVariable]

int i, j, mincost = INT_MAX, minIndex;

^

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:17:0: style: The function 'graph_create_array' is

never used. [unusedFunction]

void graph_create_array(int a[][100], int n, int e) {

^

D:\project_code\pythonproject\CodeAuditing\test_c\graph.c:160:0: style: The function 'prim_minspant' is never

used. [unusedFunction]

void prim_minspant(int GE[6][6], int n) {

^

drmemory_error : ['Error #1: LEAK 12 direct bytes 0x01495b78-0x01495b84 + 24 indirect bytes\n# 0

replace_malloc [d:\\a\\drmemory\\drmemory\\common\\alloc_replace.c:2580]\n# 1 graph_create_link

[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:43]\n# 2 main

[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:257]', 'Error #2: LEAK 36 direct bytes

0x01496fb8-0x01496fdc + 0 indirect bytes\n# 0 replace_calloc

[d:\\a\\drmemory\\drmemory\\common\\alloc_replace.c:2620]\n# 1 graph_dfs

[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:91]\n# 2 main

[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:261]', 'Error #3: LEAK 36 direct bytes

0x01497000-0x01497024 + 0 indirect bytes\n# 0 replace_calloc

[d:\\a\\drmemory\\drmemory\\common\\alloc_replace.c:2620]\n# 1 graph_bfs

[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:153]\n# 2 main
[D:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c:263]']
drmemory_summery : [['unaddressable access(es)', '0', '0'], ['uninitialized access(es)', '0', '0'], ['invalid heap
argument(s)', '0', '0'], ['GDI usage error(s)', '0', '0'], ['handle leak(s)', '0', '0'], ['warning(s)', '0', '0'], ['leak(s)', '3', '11',
'336'], ['possible leak(s)', '0', '0', '0']]
clangevaluationtext : ['', 'Filename Regions Missed Regions Cover Functions Missed Functions Executed Lines
Missed Lines Cover Branches Missed Branches Cover\n--------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------\nD:\\project_code\\pythonproject\\CodeAuditing\\test_c\\graph.c
107 40 62.62% 12 3 75.00% 213 59 72.30% 74 28 62.16%\n--------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------\nTOTAL 107 40 62.62% 12 3 75.00% 213 59 72.30% 74 28
62.16%\n']
Flawfindertext : Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining D:\project_code\pythonproject\CodeAuditing\test_c\graph.c
FINAL RESULTS:
ANALYSIS SUMMARY:
No hits found.
Lines analyzed = 265 in approximately 0.01 seconds (44156 lines/second)
Physical Source Lines of Code (SLOC) = 226
Hits@level = [0] 7 [1] 0 [2] 0 [3] 0 [4] 0 [5] 0
Hits@level+ = [0+] 7 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0
Hits/KSLOC@level+ = [0+] 30.9735 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0
Minimum risk level = 1
There may be other security vulnerabilities; review your code!
```

```
See 'Secure Programming HOWTO'

(https://dwheeler.com/secure-programs) for more information.

copy_right : Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler. and Software College of Northeastern

University

detect_rules : Number of rules (primarily dangerous function names) in C/C++ ruleset: 222

examing_file : Examining D:\project_code\pythonproject\CodeAuditing\test_c\graph.c

final_results :

analysis_summary : No hits found.

Lines analyzed = 265 in approximately 0.01 seconds (44156 lines/second)

Physical Source Lines of Code (SLOC) = 226

Hits@level = [0] 7 [1] 0 [2] 0 [3] 0 [4] 0 [5] 0

Hits@level+ = [0+] 7 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0

Hits/KSLOC@level+ = [0+] 30.9735 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0

Minimum risk level = 1

There may be other security vulnerabilities; review your code!

See 'Secure Programming HOWTO'

(https://dwheeler.com/secure-programs) for more information.

hits : No hits found.

detect_lines : Lines analyzed = 265 in approximately 0.01 seconds (44156 lines/second)

detect_real_lines : Physical Source Lines of Code (SLOC) = 226

Minimum_risk_level : Minimum risk level = 1

levels : ['7', '0', '0', '0', '0', '0']

levels_plus : ['7', '0', '0', '0', '0', '0']

levels_plus_KSLOC : ['30', '0', '0', '0', '0', '0']

Process finished with exit code 0
```