Domaći zadatak broj 4

Ime i prezime studenta: Nastasija Stankovic Broj indeksa: 17955

Uputstvo: 1. Pre početka izrade promenite ime datoteke u Domaci4_ime_prezime. (ubacite svoje ime i prezime) 2. Popunite ćeliju ispod naslova odgovarajućim podacima. 3. Za rešavanje zadataka, ukoliko je potrebno, otvorite ispod teksta zadataka dodatne ćelije za upisivanje tekstualnog odgovora (Markdown) ili programskog koda (Code). 4. Nakon završetka izrade rešenja Notebook dokument sačuvati u pdf formatu i proslediti ga nastavniku. To možete da uradite ili kroz Teams ili na mail adresu jovana.dzunic@elfak.ni.ac.rs

Zadatak 1. Dat je vektor $v \in \{-1, 1\}^{30}$, tj. vektor dimenzije 30 čije su jedine moguće vrednosti -1 i 1. Pokrenuti donji kod za generisanje jednog takvog slučajnog vektora.

In [1]:
```
import numpy as np
import numpy.random as rndm
```

In [2]:
```
v=np.array([rndm.choice([-1,1]) for i in range(30)])
print(v)
```

```
[-1  1 -1 -1  1 -1  1  1  1 -1  1 -1 -1  1  1 -1  1  1  1 -1 -1 -1 -1  1
  1 -1 -1  1  1 -1]
```

a) Koristeći indeksiranje logičkim izrazima, ili logičke nizove, napisati kod kojim se određuje ukupan broj komponenti vektora v čija je vrednost jednaka 1.

(2 poena)

In [5]:
```
print(np.count_nonzero(v==1))
```

```
15
```

b) Napisati kod kojim se ispituje da li se u prvih 8 komponenti vektora v nalaze sve jedinice.

(2 poena)

In [6]:
```
print(np.all(v[:9]==1))
```

```
False
```

c) Napisati programski kod kojim se proverava da li se u vektoru v jedinice nalaze samo u prvih 15 komponenti.

(2 poena)

In [7]:
```
a=v[16:]
print(not np.any(a))
```

```
False
```

Zadatak 2. a) Šta je izlaz sledećeg koda?

```
pythone
lista= [3, 2, 1, 4, 7, 16, 8, 10]
min_vr = 0
for broj in lista:
   if broj < min_vr:
       min_vr = broj
print(min_vr)
```

- [ ] 16
- [ ] 1
- [X] 0
- [ ] Poruka greške

(2 poena)

b) Šta je izlaz sledećeg koda?

```
pythone
m= int(input("Unesite ceo broj m:"))
n= int(input("Unesite ceo broj n:"))
if m<n:
    a=n
    b=m
else:
    a=m
    b=n
while b!=0:
   a, b=b, a%b
print(a)
```

- [X] NZD(m,n)
- [ ] NZS(m,n)
- [ ] m
- [ ] n
- [ ] max{m,n}
- [ ] Poruka greške

(2 poena)

Zadatak 3. (Arhimedova aproksimacija broja π) Oko 250. godine pne starogrčki matematičar Arhimed izveo je geometrijski algoritam za aproksimaciju broja π. Uspeo je da pokaže nejednakost $3\frac{10}{71} < \pi < 3\frac{1}{7}$. Razmotrićemo njegov postupak.

Neka je data kružnica poluprečnika $\frac{1}{2}$, tj. prečnika 1. Obim te kružnice je

$$O = 2r\pi = \pi.$$

Posmatramo niz upisanih i opisanih pravilnih mnogouglova oko date kružnice. Kako se broj temena mnogouglova povećava, tako će ovi mnogouglovi sve više ličiti na kružnicu. Ukoliko označimo S_m obim upisanog pravilnog $m-$to ugla, a O_m obim opisanog, važiće

$$S_m < O < O_m, \qquad \forall m \in \mathbb{N}.$$

Videti sliku, tj. aktivirati naredni kod.

In [8]:
```
from ipywidgets import interactive
import matplotlib.pyplot as plt

def mnogouglovi(m):
    t=np.linspace(0,2*np.pi,m+1)
    x_upisan=0.5*np.cos(t)
    y_upisan=0.5*np.sin(t)
    r=0.5/np.cos(np.pi/m)
    x_opisan=r*np.cos(t)
    y_opisan=r*np.sin(t)
    plt.plot(x_upisan,y_upisan,'b')
    plt.plot(x_opisan,y_opisan,'g')
    theta=np.linspace(0,2*np.pi,50)
    x_krug=0.5*np.cos(theta)
    y_krug=0.5*np.sin(theta)
    plt.plot(x_krug,y_krug,'k')
    plt.title("Arhimedova aproksimacija broja $\pi$")
    plt.axis('equal')
    plt.xlim(-1,1)
    plt.ylim(-1,1)

interaktivna_slika = interactive(mnogouglovi, m=(3,25, 1))
interaktivna_slika
```

Ukoliko označimo $a_n = O_{2^n}$ i $b_n = S_{2^n},\ n \geq 2$ za njih važe rekurentne formule:

$$a_{n+1} = \frac{2a_n b_n}{a_n + b_n}, \qquad b_{n+1} = \sqrt{a_{n+1} b_n}, \qquad n \geq 2.$$

Napisati program kojim će se generisati članovi nizova a_n i $b_n,\ n = 1, 2, \ldots,$ sve dok ne bude ispunjen uslov

$$|a_n - b_n| < 1e - 10.$$

Prikazati dobijenu aproksimaciju broja $\pi \approx \frac{a_n + b_n}{2}$ sa 8 decimalnih mesta.

Ukoliko vas zanima izvođenje ovih formula pogledajte dodatak.

(10 poena)

In [9]:
```
import numpy as np
a=2*np.sqrt(2)
b=4
while (abs(b-a)>1e-10):
    m=2*b*a/(b+a)
    k=np.sqrt(m*a)
    b=m
    a=k
print ("{:.8f}".format((a+b)/2))
```

```
3.14159265
```

Zadatak 4. Dat je skup od n različitih centara (tačaka) $c_1, c_2, \ldots, c_n \in \mathbb{R}^m$. Neka je S skup svih tačaka iz $\mathbb{R}^m$ koje su najbliže centrima $c_1, \ldots, c_k$, tj.

$$S = \left\{ v \in \mathbb{R}^m \mid \|v - c_i\| \leq \|v - c_j\|,\ \forall i = 1, \ldots, k,\ \forall j = k+1, \ldots, n \right\}.$$

Dokazati da je S konveksan skup koristeći svojstvo preseka konveksnih skupova. Podrazumeva se da je $\|\cdot\|$ Euklidova norma.

(10 poena)

Posmatranjem jedne kombinacije indeksa i, j, može se na osnovu uslova zadatka uvideti sledeće:

$$||v - c_i|| \leq ||v - c_j||,$$

gde je $v = (a, b, c. \ldots), c_i = (a_i, b_i, c_i. \ldots), c_j = (a_j, b_j, c_j. \ldots)$

Kvadriranjem ovog izraza dobijamo:

$$||v - c_i||^2 \leq ||v - c_j||^2$$

Na osnovu ovoga, sledi: $a^2 - 2a \cdot a_i + a_i^2 + b^2 - 2b \cdot b_i + b_i^2 + \ldots \leq a^2 - 2a \cdot a_j + a_j^2 + b^2 - 2b \cdot b_j + b_j^2 + \ldots$

Daljim sređivanjem, dobijamo:

$2(a_j - a_i)a + 2(b_j - b_i)b + \ldots \leq k_i j$

$(a_j - a_i)a + (b_j - b_i)b + \ldots \leq \alpha i_j$

Za neko i i j dobijamo formulu kojom možemo odrediti skup hiper-ravni koje dele R^m na pola. Možemo dokazati na sledeći način: $v_3 = (1 - \lambda)v_1 + \lambda v_2,$

$\lambda \in [0, 1]$

Uzimamo za vektore v_1, v_2 da su u skupu hiper-ravni za neko i,j. Neka je a_1 je komponenta a kod vektora v_1, b_1 je komponenta b kod vektora v_1

$(a_j - a_i)a_1 + (b_j - b_i)b_1 + \ldots \leq a_{ij}$

i a_2 je komponenta a kod vektora v_2, b_2 je komponenta b kod vektora v_2

$(a_j - a_i)a_2 + (b_j - b_i)b_2 + \ldots \leq a_{ij}$

Ukoliko ovo primenimo na vektor v_3 dobijamo sledeće: $(a_j - a_i)a_3 + (b_j - b_i)b_3 + \ldots = (1 - \lambda)((a_j - a_i)a_1 + (b_j - b_i)b_1 + \ldots) + \lambda((a_j - a_i)a_2 + (b_j - b_i)b_2 + \ldots)$

Članovi uz $(1 - \lambda)$ i λ mogu da budu samo a_{ij}, pa na osnovu toga sledi:

$(a_j - a_i)a_3 + (b_j - b_i)b_3 + \ldots = (1 - \lambda)a_{ij} + \lambda a_{ij} = a_{ij}$

To je maksimalna vrednost izraza za vektor v_3, ali može biti i manja od a_{ij}. Ovim smo pokazali da je za neke vrednosti i,j skup hiper-ravni konveksan skup.

Da bi skup S bio konveksan, presek svih skupova i,j mora da bude konveksan, a kako su svi skupovi i,j u našem slučaju konveksni, možemo zaključiti da je i skup S konveksan.

Zadatak 5. (Izračunavanje inercije kroz iteracije)

Jedan od osnovnih problema prilikom razvoja i implementacije algoritama numeričke linearne algebre jeste njihovo testiranje korektnosti. Naime, algoritmi imaju svojstvo da rade baš ono što je programirano, čak i ako to nije ono što smo planirali! Zbog toga je potrebno dizajnirati testove pomoću kojih se analizira način rada programskog koda i postojanje bagova u njemu. Takve veštačke uslove provere rada i logičkog dizajna algoritma nazivamo test primerima. Planiranje test primera predstavlja svojevrstan problem za sebe.

Za algoritme klasterizacije, i njima srodne druge algoritme, Python modul Scikit learn poseduje podršku za generisanje skupova podataka pogodnih za testiranje

algoritama.

Naredba make_blobs() kreira podatke zadatog obima i dimenzije sa karakterisitikama klastera koje želimo. Elementi pojedinačnih klastera imaju karakteristike normalne raspodele svojih koordinata. Parametri naredbe `make_blobs` kojima se određuje izgled podataka (i njihove podrazumevane vrednosti) su:

- `n_samples` **= 100, predstavlja ukupan broj podataka ukoliko je unešena jedna vrednost. Kada je unos niz ili lista, ona predstavlja broj podataka za svaki od pojedinačnih klastera;**
- `n_features` **= 2, dimenzionalnost podataka;**
- `centers` **= None, broj centara ili fiksirane koordinate centara;**
- `cluster_std` **= 1.0, standardne odstupanje elemnata unutar klastera od centara;**
- `center_box` **= -10.0, 10.0, opseg koordinata centara klastera kada se oni zadaju slučajnim izborom;**
- `shuffle` **= True, mešanje podataka prilikom izlaza, tj. redosled podataka nije u skladu sa klasterom kome pripada;**
- `return_centers` **= False, izlazni podaci naredbe sadrže koordinate centara generisanih klastera.**

Izlaz naredbe predstavljaju:

- X **- matrica generisanih podataka dimenzije** `n_samples` $\times$ `n_features`
- y **- vektor labela za svaki od generisanih podataka**
- c **- niz koordinata generisanih centara dimenzije** `n_centers` $\times$ `n_features`**. Postoji kao izlaz samo kada je** `return_centers` **= True.**

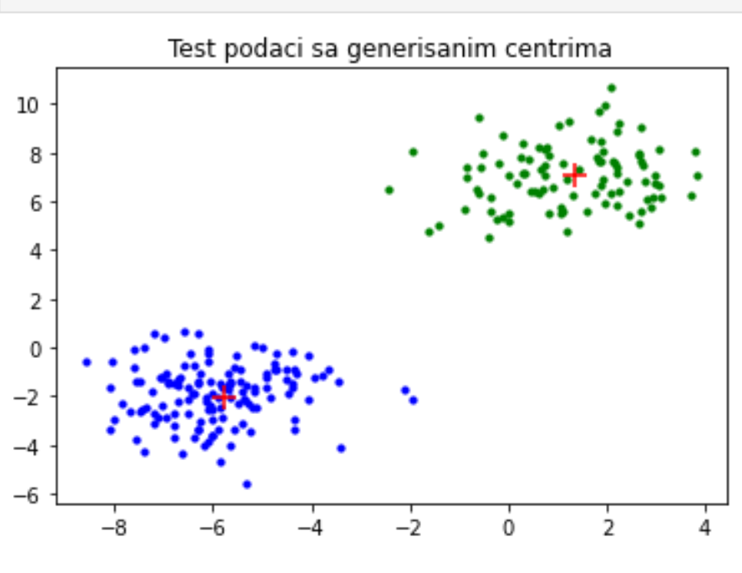
Za naredni zadatak test primer skupa dvodimenzionalnih podataka kreiran je naredbom `make_blob`. Podaci su podeljeni u dva klastera od 132 i 95 elemenata.

In [15]:
```
from sklearn.datasets import make_blobs

m=np.array([132,95])

tacke, labele, centri = make_blobs(n_samples=m, n_features=2, cluster_std=[1.2,1.3], return_centers=True)
print(tacke.shape)
print(labele.shape)
print(centri.shape)
print(centri)
print(labele[:20])
```

```
(227, 2)
(227,)
(2, 2)
[[-5.77059298 -2.080561  ]
 [ 1.35440717  7.08032432]]
[0 0 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0]
```

Grafički prikaz test-podataka dobija se sledećim naredbama.

In [16]:
```
grupa1=tacke[labele==0]
grupa2=tacke[labele==1]
plt.scatter(grupa1[:,0],grupa1[:,1],c='b', s=10)
plt.scatter(grupa2[:,0],grupa2[:,1],c='g', s=10)
plt.scatter(centri[:,0],centri[:,1], s=150, c='r', marker='+')
plt.title("Test podaci sa generisanim centrima");
```

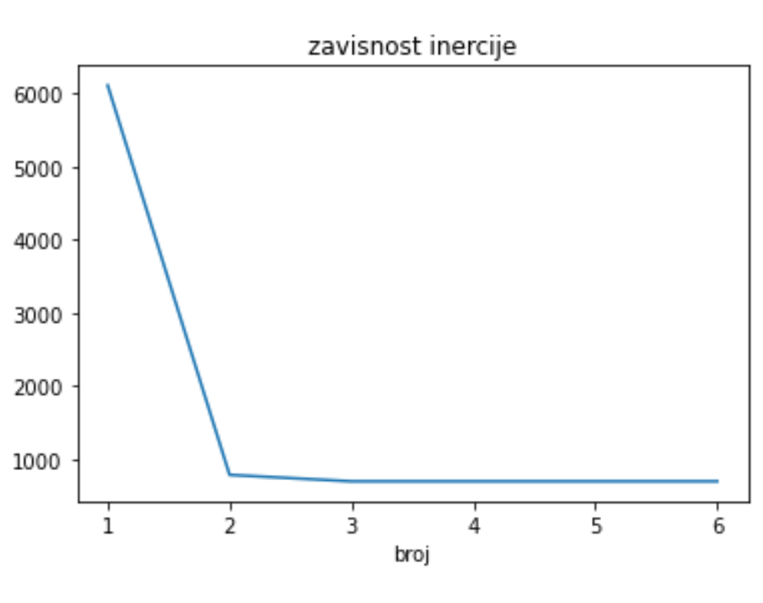


Upotrebom `for` petlje napraviti 6 iteracija K-means algoritma nad upravo kreiranim podacima `tacke`. Algoritam se sprovodi za 2 klastera sa proizvoljno izabranim početnim centrima (sami ih odaberite, ali daleko od centara klastera). U svakoj iteraciji računati koeficijent inercije. Po završetku iteracija napraviti grafik funkcije inercije u zavisnosti od broja iteracija.

(10 poena)

In [23]:
```
a=[0,0]
b=[0,0]
a[0]=np.random.uniform(-10,10)
a[1]=np.random.uniform(-10,10)
b[0]=np.random.uniform(-10,10)
b[1]=np.random.uniform(-10,10)


iner=[]
for i in range(6):
    for j in range(0,len(tacke)):
        C = (tacke[j][0]-a[0])**2 + (tacke[j][1]-a[1])**2
        D = (tacke[j][0]-b[0])**2 + (tacke[j][1]-b[1])**2
        if C < D:
            labele[j]=0
        else:
            labele[j]=1
    pom=0
    for k in range(0,len(tacke)):
        if labele[k] == 0:
            pom+=(tacke[k][0]-a[0])**2+(tacke[k][1]-a[1])**2
        else:
            pom+=(tacke[k][0]-b[0])**2+(tacke[k][1]-b[1])**2


    iner.append(pom)
    temp1=0
    temp2=0
    x1=[0,0]
    x2=[0,0]
    for k in range(0,len(tacke)):
        if labele[k]==0:
            x1[0]+=tacke[k][0]
            x1[1]+=tacke[k][1]
            temp1+=1
        else:
            x2[0]+=tacke[k][0]
            x2[1]+=tacke[k][1]
            temp2+=1

    a[0]=x1[0]/temp1
    a[1]=x1[1]/temp1
    b[0]=x2[0]/temp2
    b[1]=x2[1]/temp2

plt.plot(np.arange(1,7),iner)
plt.title("zavisnost inercije")
plt.xlabel("broj")
```

Out[23]:
```
Text(0.5, 0, 'broj')
```



Zadatak 6. (Primena KMeans i make_blobs naredbi) Kreirati test primer trodimenzionalnih podataka za 5 klastera sa 54,88,13,67 i 90 elemenata. Primeniti `KMeans` naredbu za broj klastera $k = 3, 4, \ldots, 10$. Prikazati koeficijente inercije i siluete u funkciji od broja klastera.

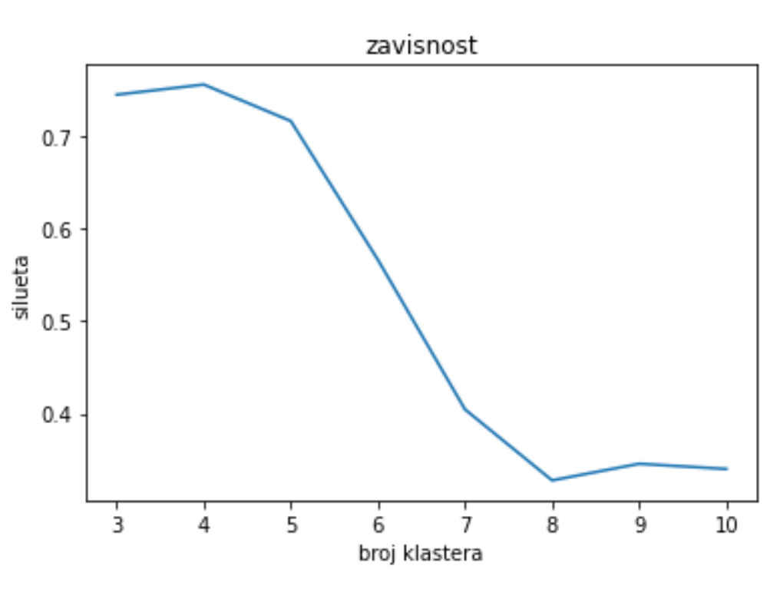
(10 poena)

In [22]:
```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import numpy as np
from sklearn.metrics import silhouette_score

niz=np.array([54,88,13,67,90])
kl=[]
sil=[]
x,y,z=make_blobs(n_samples=niz, n_features=3, return_centers=3, random_state=None)
for j in range(3,11):
    i=KMeans(n_clusters=j)
    i.fit(x)
    kl.append(i.inertia_)
    sil.append(silhouette_score(x, i.labels_))
plt.plot(np.arange(3,11),kl)
plt.title("zavisnost")
plt.xlabel("broj klastera")
plt.ylabel("inercija")
```

Out[22]:
```
Text(0, 0.5, 'inercija')
```

In [21]:
```
plt.plot(np.arange(3,11), sil)
plt.title("zavisnost")
plt.xlabel("broj klastera")
plt.ylabel("silueta")
```

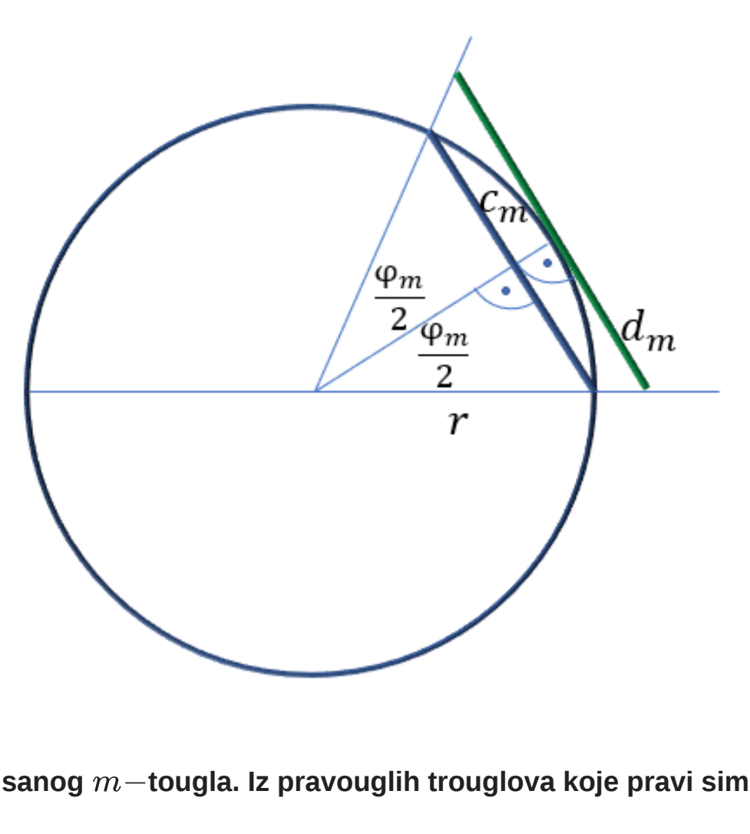
Out[21]:
```
Text(0, 0.5, 'silueta')
```



Dodatak

Objašnjenje formula upotrebljenih u zadataku 3.

Pravilni $m-$tougao ima centralni ugao (ugao sa temenom u centru opisane/upisane kružnice i kracima kroz dva susedna temena mnogougla) jednak $\varphi_m = \frac{2\pi}{m}$.



Označimo sa c_m i d_m redom dužine stranica upisanog i opisanog $m-$tougla. Iz pravouglih trouglova koje pravi simetrala centralnog ugla, za c_m i d_m važi:

$$c_m = 2r\sin(\varphi_m/2), \qquad d_m = 2r\tan(\varphi_m/2).$$

Obimi ovih mnogouglova tada su

$$S_m = mc_m = 2mr\sin(\varphi_m/2), \qquad O_m = md_m = 2mr\tan(\varphi_m/2).$$

Uvedimo oznake $m = 2^n,\ n \geq 2, \quad \frac{\varphi_m}{2} = \theta_n$ i

$$a_n = O_m = O_{2^n} = 2 \cdot 2^n r\tan\theta_n = 2^{n+1} r\tan\theta_n, \qquad b_n = S_m = S_{2^n} = 2 \cdot 2^n r\sin\theta_n = 2^{n+1} r\sin\theta_n, \tag{1}$$

$$a_{n+1} = O_{2m} = O_{2^{n+1}} = 2^{n+2} r\tan\theta_{n+1} = 2^{n+2} r\tan(\theta_n/2), \qquad b_n = S_m = S_{2^n} = 2^{n+2} r\sin(\theta_n/2). \tag{2}$$

Pokazaćemo da važi

$$a_{n+1} = \frac{2a_n\, b_n}{a_n + b_n}, \qquad b_{n+1} = \sqrt{a_{n+1} b_n}, \qquad n \geq 2.$$

$$\frac{2a_n\, b_n}{a_n + b_n} = \frac{2 \cdot 2^{n+1} r\tan\theta_n \cdot 2^{n+1} r\sin\theta_n}{2^{n+1} r\tan\theta_n + 2^{n+1} r\sin\theta_n} = 2^{n+2} r\frac{\tan\theta_n\ \sin\theta_n}{\tan\theta_n + \sin\theta_n} = 2^{n+2} r\frac{\sin\theta_n}{1 + \cos\theta_n} \tag{3}$$

$$= 2^{n+2} r\frac{2\sin(\theta_n/2)\cos(\theta_n/2)}{2\cos^2(\theta_n/2)} = 2^{n+2} r\frac{\sin\theta_{n+1}}{\cos\theta_{n+1}} = a_{n+1}, \tag{4}$$

$$\sqrt{a_{n+1} b_n} = \sqrt{2^{n+2} r\frac{\sin\theta_{n+1}}{\cos\theta_{n+1}} \cdot 2^{n+1} r\sin\theta_n} = \sqrt{2^{n+2} r\frac{\sin\theta_{n+1}}{\cos\theta_{n+1}} \cdot 2^{n+1} r \cdot 2\sin\theta_{n+1}\cos\theta_{n+1}} \tag{5}$$

$$= 2^{n+2} r\sin\theta_{n+1} = b_{n+1}. \tag{6}$$